Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Estructuras de Datos
Proyecto 1
M.Sc Luis Espino Aux: Andree Avalos
Ing. Jesús Guzmán, Aux: Marvin Martinez
Ing. Alvaro Hernandez Aux: Juan Landaverde

PSEUDO-CAD

Descripción General

La importancia de la simulación en problemas reales de aplicación ha tomado un gran auge, y por lo tanto, se ha hecho importante el desarrollo de este tipo de software, pues las empresas lo utilizan para ver el comportamiento de los sistemas a lo largo del tiempo y sacar conclusiones de esta observación, ahorrando tiempo y recursos al no utilizar el sistema real para realizar las pruebas.

Pseudo-Cad es una aplicación que le permite al usuario diseñar interiores complejos cargando sus propias librerías. El programa es capaz de soportar múltiples proyectos los cuales están conformados por diversos niveles que representan un piso de la estructura que se desea simular. El diseño de interiores incluye que el usuario pueda colocar múltiples objetos y la posibilidad de rotar los objetos en diferentes grados dando una mayor personalización al interior.

Índice

Objetivos

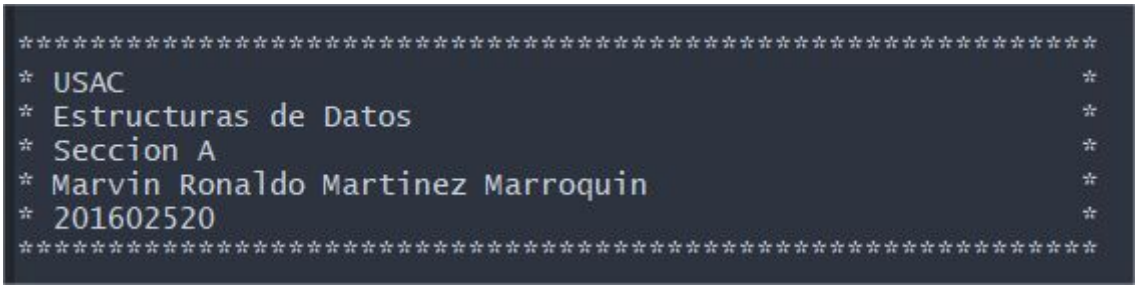
- Que el estudiante se familiarice con el lenguaje C++
- Que el estudiante sepa envolverse en el ámbito de una simulación
- Comprender y desarrollar distintas estructuras de datos no lineales como lo son matrices dispersas y arboles binarios.
- Definir algoritmos de búsqueda y recorrido.
- Familiarizarse con la herramienta Graphviz para la generación de reportes de estructuras gráficos. 7. Familiarizarse con el manejo
- Familiarizarse con el manejo de lectura de archivos JSON.

Descripción del proyecto

Menú principal

Al iniciar con la ejecución del programa se debe mostrar un encabezado con los datos del estudiante.

- Nombre del curso
- Sección
- Nombre del estudiante
- Número de carnet



Luego se debe mostrar las múltiples opciones para que el usuario seleccione una de ellas ingresando el número correspondiente.

- Ver Proyectos
- Editar Proyectos
- Cargar Proyecto
- Graficar Proyectos
- Guardar Proyectos
- Cargar Librerías

Ver Proyectos

En esta sección del programa se deberá listar todos los proyectos que existan en la estructura de almacenamiento(**ARBOL AVL**).

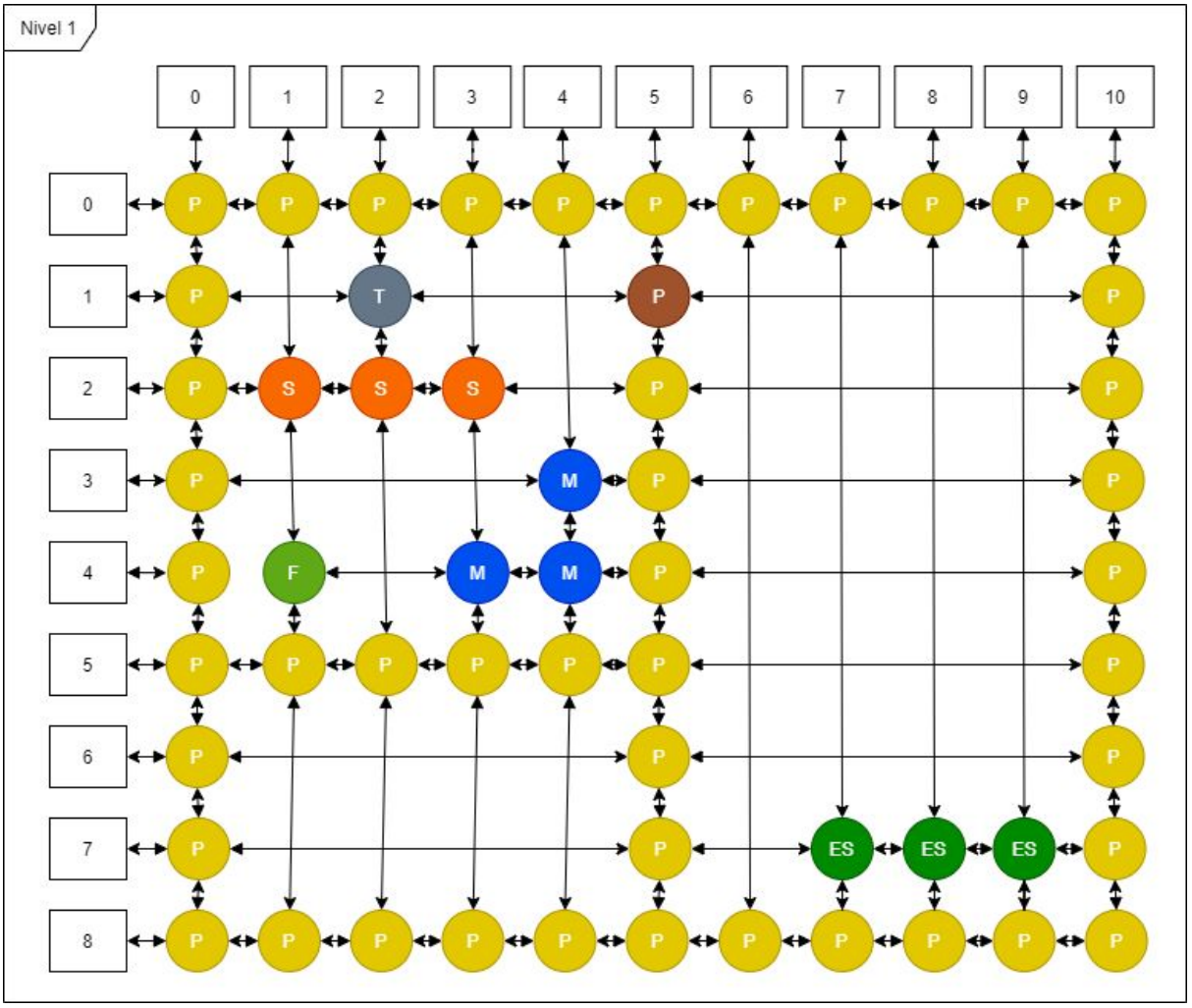
Ejemplo:

```
1. Proyecto 1
2. Proyecto 2
3. Proyecto 3
4. Proyecto 4
5. Proyecto 5
6. Proyecto 6
7. Proyecto 7
```

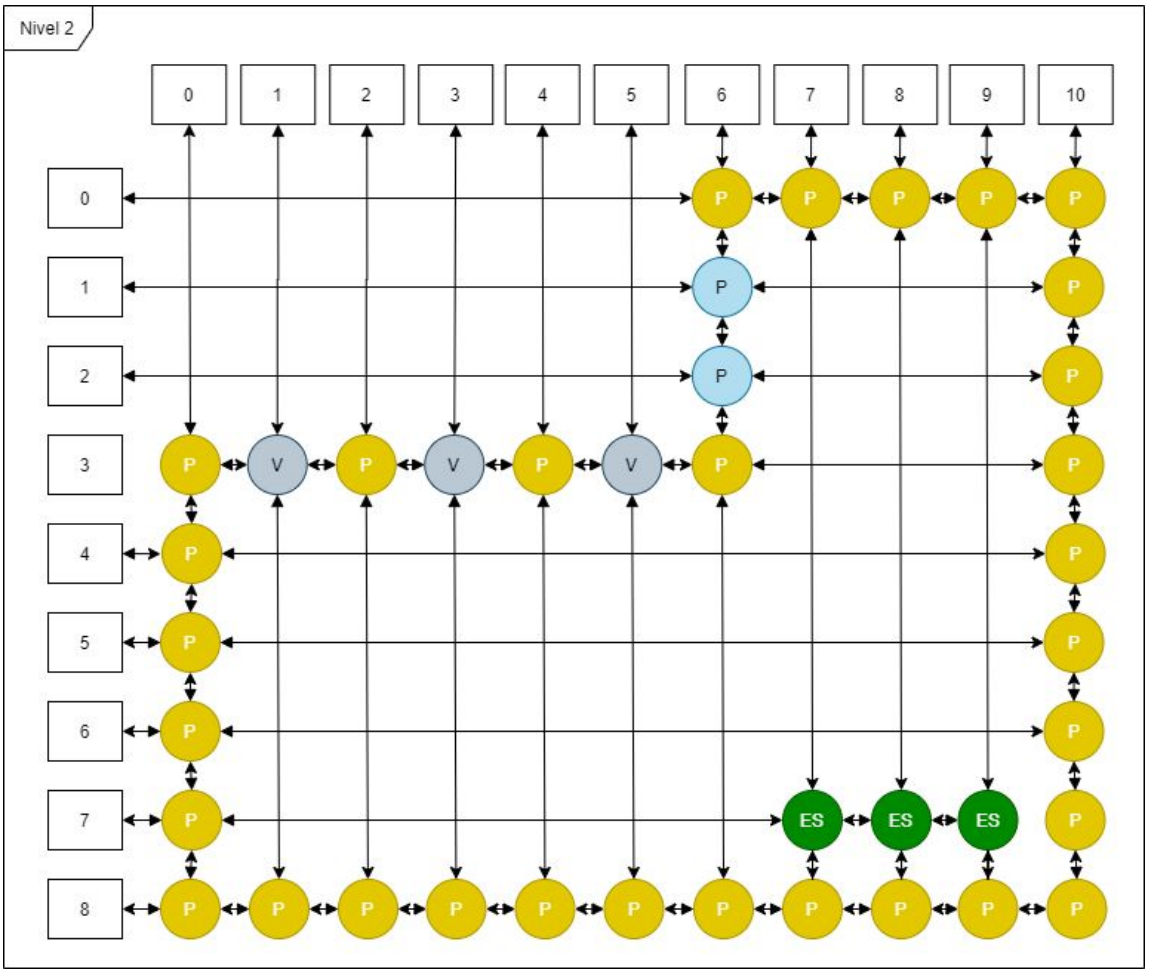
Al seleccionar un proyecto este procederá a graficar todos los niveles que contiene el proyecto (**Matriz Dispersa)** de forma gráfica con la herramienta graphviz.

Ejemplo:

Nivel 1



Nivel 2



Editar Proyecto

El programa tendra la opcion de poder editar el nivel, así mismo poder eliminar el nodo del **árbol AVL.**

Ejemplo:

Agregar Nivel

Cuando el usuario seleccione agregar nivel se tendrá que pedir la direccion de donde se encuentre el archivo JSON a cargar con toda la información del nivel.

En el siguiente enlace se puede encontrar un ejemplo de los niveles masivos del programa.
https://drive.google.com/file/d/1xbi7WVEXx9zt-jiiKYbMrBMZepnWvYm7/view

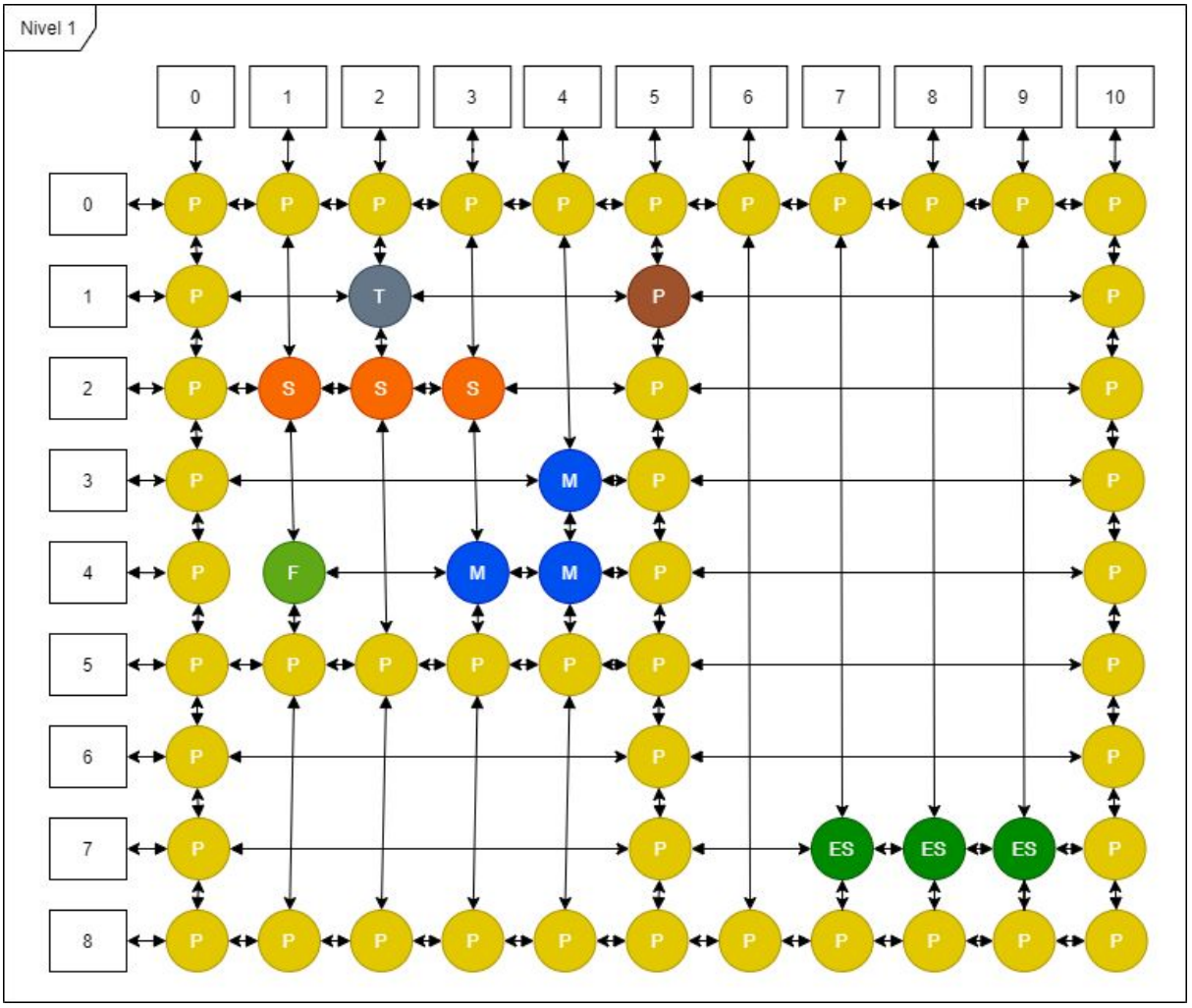
Editar Nivel

Si el usuario selecciona la opción editar nivel se abrirá un submenú el cual contendrá un listado de niveles actuales del proyecto, cuando el usuario seleccione el nivel se deberá mostrar de forma gráfica(**Graphviz)** como esta distribuido este nivel en la **matriz dispersa**, a su vez se abrirá un submenú para poder agregar o eliminar objetos; estos objetos están cargados en una estructura de datos tipo **árbol binario de búsqueda.**

Ejemplo de menú de niveles :

```
**********************
* 1. Nivel 1         *
* 2. Nivel 2         *
**********************
```

Ejemplo de Nivel 1:



Ejemplo de Menú Objetos:

```
1. Agregar objeto
2. Eliminar objeto
3. Eliminar Pared
```

Copiar Nivel

Esta opción le permite al usuario hacer una copia de el nivel el que seleccione, el usuario tiene que ingresar el número del nivel que se quiere copiar luego el programa le preguntará en qué nivel quiere copiarlo. Si el nivel al cual se quiere copiar los objetos no está vacío se tiene que eliminar el contenido del nivel y después se cargan los objetos.

Crear Cantidad de pisos

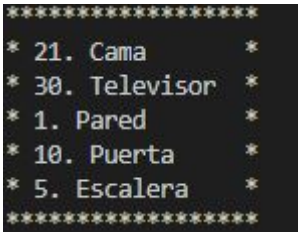
Esta opción le permite al usuario crear cierta cantidad de pisos que el usuario necesite. Primero se pregunta cuántos pisos quiere y se crean a partir del último nivel, estos niveles no tendrán objetos (mayor referencia en: Flujo principal del proyecto)

En el siguiente enlace se puede encontrar un ejemplo de los niveles masivos del programa.
https://drive.google.com/file/d/1xbi7WVEXx9zt-jiiKYbMrBMZepnWvYm7/view

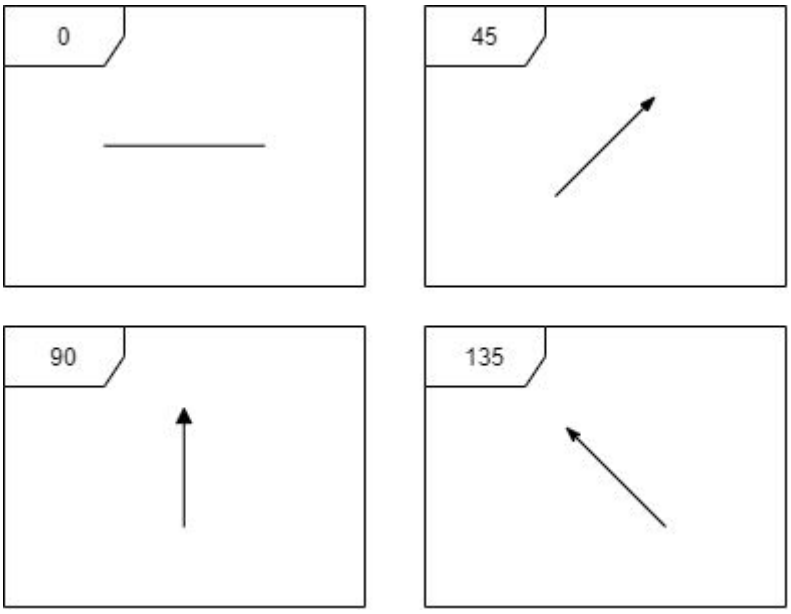
Agregar objeto

En este submenú se mostrarán todos los objetos disponibles del árbol binario de búsquedas y así poder elegir uno. Se tiene que ingresar el id del objeto, la fila, columna en donde se quiere insertar y los grados de la posición de la figura.

Ejemplo de objetos en ABB:



Ejemplo de número en grados permitidos:



Ángulos admitidos: 0,45,90,135,180,225,270,315

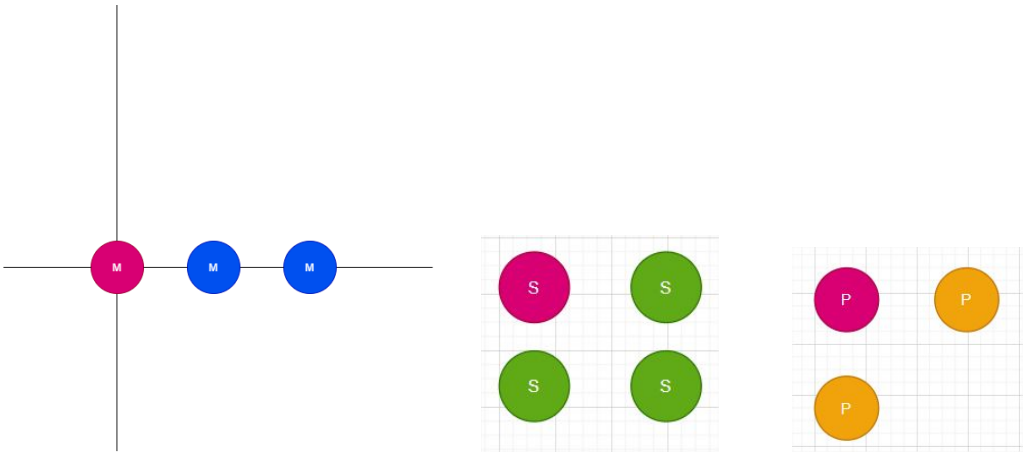
En el siguiente enlace se puede encontrar un ejemplo de las librerías del programa.
https://drive.google.com/file/d/1NbMcRqSAdfl6G3CJiSiEfJMmhbxEzMSA/view

Grados de un objeto

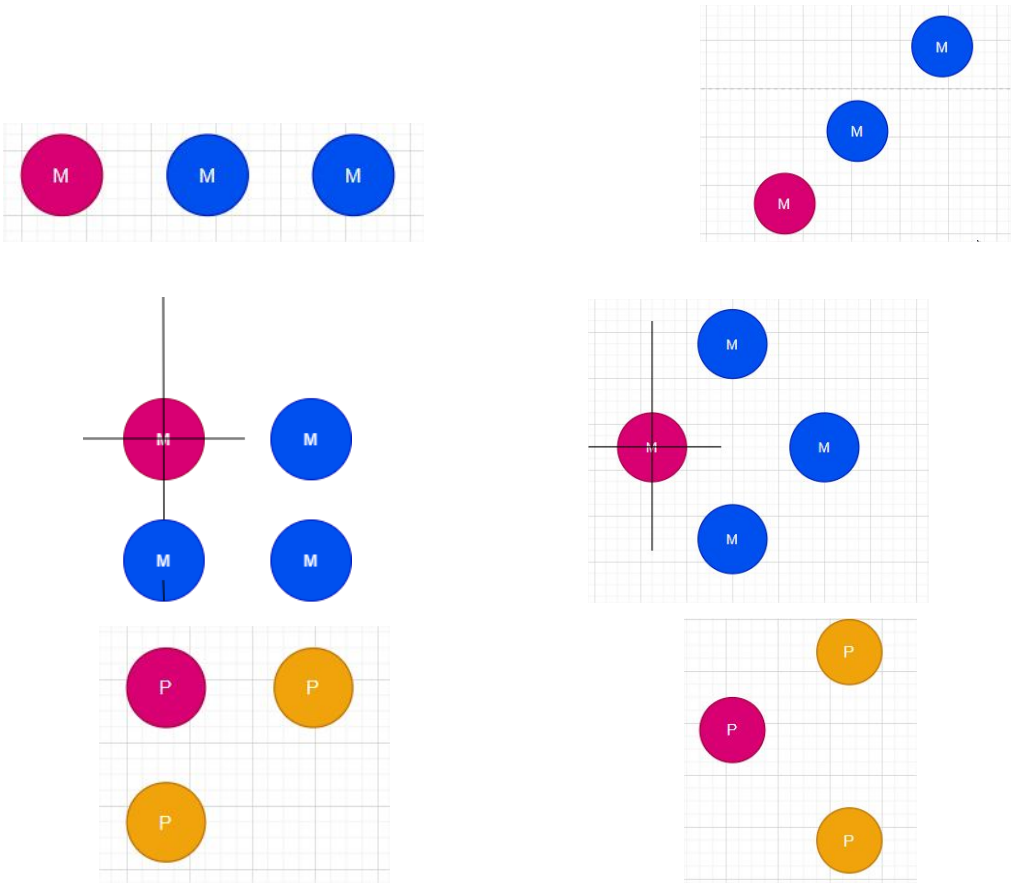
Cuando un objeto es posicionado por la librería se le puede indicar el número de grados a los que quiere ser colocado permitiendo una mayor personalización de la herramienta.
Para rotar un objeto necesitamos utilizar un eje de referencia para que la figura siempre gire de la misma manera, también el eje nos ayudará para hacer los cálculos si el objeto no se está colocando sobre algún objeto que no permite colocarlo de esa manera.

La regla para identificar el eje de la figura es el nodo que se encuentre más a la izquierda y más arriba de los nodos del objeto.

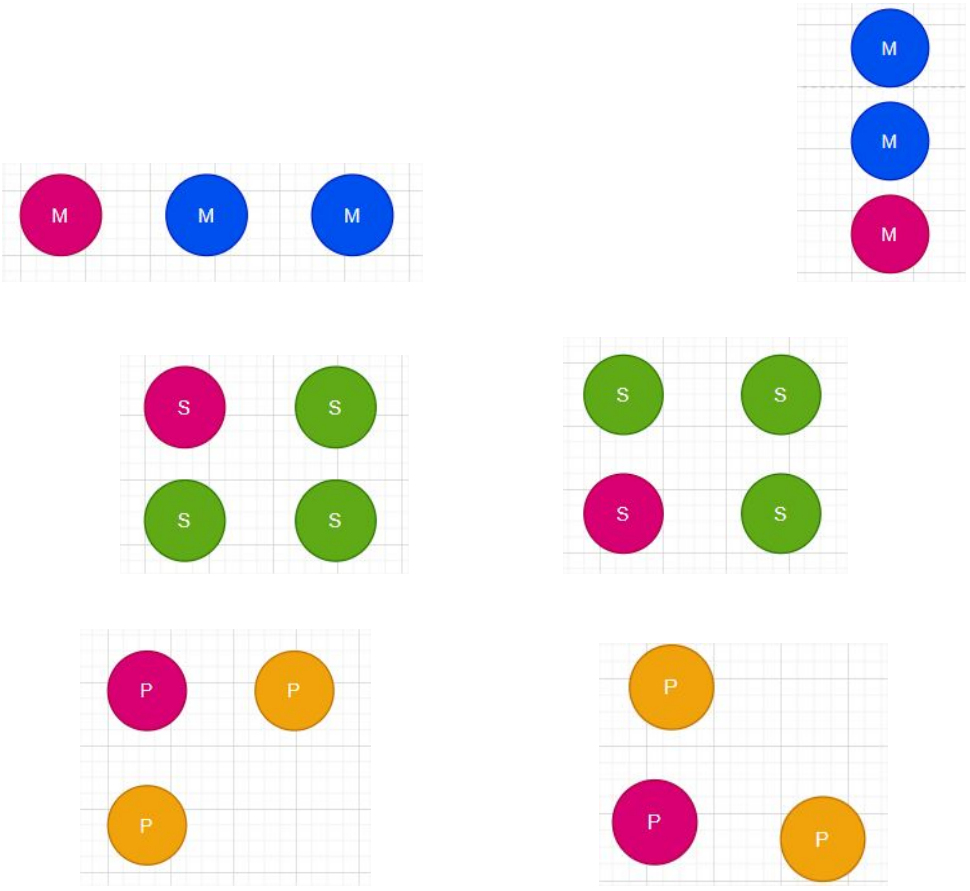
La figura siempre tiene que ser girada de su forma base que es la forma en la que está en la librería. Los siguientes imagenes serian la forma base de los objetos y el nodo rojo sería su vértice aplicando la regla.



Una rotación de 45 grados para la siguiente figura seria de la siguiente manera.



Un algoritmo para rotar 90 grados podría ser primero aplicar una rotación de 45 y luego rotar esa figura otros 45 grados.



En caso de que una figura tenga colisión se debe imprimir un mensaje que le indique al usuario que no es permitido guardarlo en esa posición y retirar la figura que se está colocando.

Mover Objeto

Esta función se le mostrará un listado de los objetos que se encuentran en ese nivel, luego el puede seleccionar un objeto y tiene que ingresar la nuevas coordenadas en las que quiere colocar el objeto.

Eliminar Objeto

Para la sección eliminar objetos se tendrán que mostrar listados todos los objetos de ese nivel, entonces el usuario tendra la opcion de elegir uno y los nodos relacionados a este objeto se tendrán que eliminar de el **árbol binario de busqueda** asi como sus elementos en la **matriz dispersa**.

Ejemplo de objetos agregados:

```
**********************
* 21. Cama           *
* 30. Televisor      *
* 1. Pared           *
* 10. Puerta         *
* 5. Escalera        *
**********************
```

Eliminar Pared

Esta sección estará dentro de la sección de nivel, y éste será capaz de eliminar un rango de paredes, se deberá pedir si desea ingresar coordenada, confirmar para poder eliminar las coordenadas o cancelar la eliminación.

```
1. Eliminar
2. Confirmar
3. Cancelar
```

Si elige eliminar paredes entonces el programa le pedirá las coordenadas de la pared, permitiendo al usuario ingresar muchas coordenadas para eliminar por grupo, si el dato no es una coordenada como por ejemplo una letra deberá mostrar el siguiente mensaje.

```
Continuar Y/N
```

Si le da a la opción " Y/y" entonces pedirá nuevamente coordenadas
de caso contrario esto terminará y volverá al menú anterior

Ejemplo de coordenadas:
1,2
1,1
1,0

todas las coordenadas se mantendrán en memoria hasta cancelar o confirmar.
Si el usuario elige confirmar entonces el programa empezará a eliminar cada una de las coordenadas que están en memoria, si le da la opción cancelar no se ejecuta la acción y las coordenadas tendrán que borrarse de la memoria dinámica.
Cada vez que se agregue o elimine un objeto la imagen del nivel se debe generar automáticamente para poder visualizar los cambios.

Eliminar Nivel

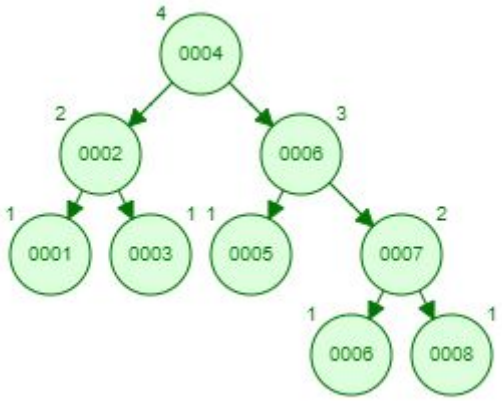
Si el usuario elige esta opción se listaran los niveles que posee el proyecto, para su posterior eliminación de la **matriz dispersa** y también debe eliminarse su **árbol binario de búsqueda** de objetos.

```
******************
* 1. Nivel 1     *
* 2. Nivel 2     *
******************
```

Eliminar Proyecto

Cuando el usuario seleccione esta opción en el submenú, se debe eliminar el proyecto (**nodo de árbol AVL**).

Ejemplo de árbol AVL:



Cargar Proyectos

En esta sección de la aplicación se podrá cargar los proyectos en un **árbol AVL** es equivalente a un edificio. Cada proyecto tendrá su definición de sus espacios en múltiples niveles, estos nodos del **árbol AVL** estarán representados por una matriz dispersa multidimensional y un **árbol binario de búsqueda** que indicará los objetos de cada nivel.

En el siguiente enlace se puede encontrar un ejemplo de los niveles masivos del programa.
https://drive.google.com/file/d/18YkZVnS-PENU6EkJepGfv-h8xSo0-LAZ/view

Guardar Proyectos

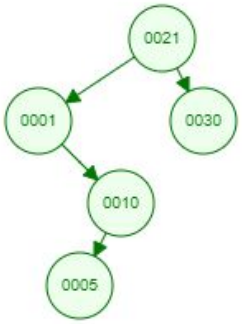
Esta opción les permite generar un archivo JSON idéntico a los archivos de entrada para guardar el progreso que se realizó en la aplicación.

Cargar Librerías

En esta sección el programa tendrá la capacidad de cargar librerías, este con el objetivo de tener objetos predefinidos para un mejor entendimiento de este, para esto los objetos tendrán los siguientes atributos:

- Identificador único
- Nombre
- Letra representativa
- Color
- Puntos ej:(1,1)(1,2)(1,3)

Estos deben ser almacenados en un árbol binario de búsqueda.



Si se hace una carga masiva y el objeto no existe dentro de las librerías el programa tiene que agregarlas a sus librerías para volver a utilizarlos.

Reportes

En esta sección el estudiante deberá ser capaz de dar detalle de cada uno de los siguientes enunciados.

1. Mostrar árbol avl de forma gráfica con los proyectos actuales.
2. Mostrar árbol binario de búsqueda con los objetos cargados provenientes de las librerías.
3. Mostrar de forma ordenada los proyectos con mayor número de niveles de forma descendente
4. Mostrar de forma ordenada los proyectos con mayor número de niveles de forma ascendente
5. En un proyecto mostrar niveles de forma ordenada por el número de objetos de menor a mayor.
6. En un proyecto mostrar el nivel que tenga más espacio.
7. En un proyecto mostrar el nivel que tenga menos paredes.
8. En un proyecto mostrar el nivel que tenga más paredes.
9. En un proyecto mostrar el nivel que tenga más espacio y más ventanas.
10. En un proyecto mostrar el nivel que tenga más espacio y menos ventanas.

Flujo principal del proyecto

En este apartado explicaremos de mejor manera como debe comportarse el proyecto.

- El estudiante debe hacer pull en el último commit hecho, el cual debe ser antes de la hora y fecha máxima de entrega.
- El estudiante debe proceder a abrir la aplicación descargada anteriormente.
- El programa debe mostrar el encabezado y el menú principal en una terminal.
- Se procederá a cargar las librerías de objetos, para esto se va a la opción **cargar librerías** del menú principal, y luego ingresa la ruta del archivo proporcionado por los auxiliares, estos objetos se almacenarán en un **árbol binario de búsqueda global.**
- Luego de haber cargado los objetos que se pueden utilizar dentro de PSEUDO-CAD, se procederá a cargar los proyectos, para esto el estudiante debe ir al menú principal y seleccionar la opción **cargar proyectos**.
- Deberá pedir la ruta json donde se encuentran todos los proyectos y proceder a leerlos.
    - Se creará un **árbol avl** donde su índice será el nombre del proyecto basado en la suma de sus caracteres convertidos a código **ASCII** (“casa 1” = 99 + 97 + 115 + 97 + 32 = 440)
    - Por cada nodo **arbol avl** tendrá una **estructura** (queda a discreción del estudiante) el cual en sus nodos tendrá entre sus atributos:
        - **Una matriz dispersa**, la cual antes de colocar cada nodo tendrá que hacer una validación de cada objeto que se quiera poner en sus nodos matriz, exceptuando los objetos pared, estos objetos al no estar en la librería de objetos solo deben ser colocados, también el programa debe ser capaz de saber si un nodo está ocupado o no, en el caso uno de los nodos donde se quiere colocar el objeto está ocupado este objeto **no** se sobrepondra al que ya esta puesto, se tendrá que ignorar y seguir con los demás (esto vale para pared y objetos).
        - **Un árbol binario** de búsqueda donde se almacenarán los objetos, por ejemplo si el nivel uno solo tiene una televisión esta deberá estar colocada según la cantidad de sus nodos en la matriz y también debe estar registrada en el árbol binario de búsqueda (sólo su identificador, color, su descripción y una lista de coordenadas por cada nodo del objeto)
        - Además de descripcion, fecha de creación, etc...
        - **Cada nodo pivote debe apuntar hacia el siguiente nivel si en dado caso existiera más de uno (utilizando una estructura de datos adecuada)**
        - **Se tendrá una matriz de nivel y un abb por cada nodo pivote.**

- Se procederá a mostrar todos los niveles del proyecto en la sección de menú principal
- Los nodos de la **matriz deben estar coloreados** con su color correspondiente, y el **nombre que deben mostrar es la primera inicial del nombre del objeto**.

Diccionario de Términos

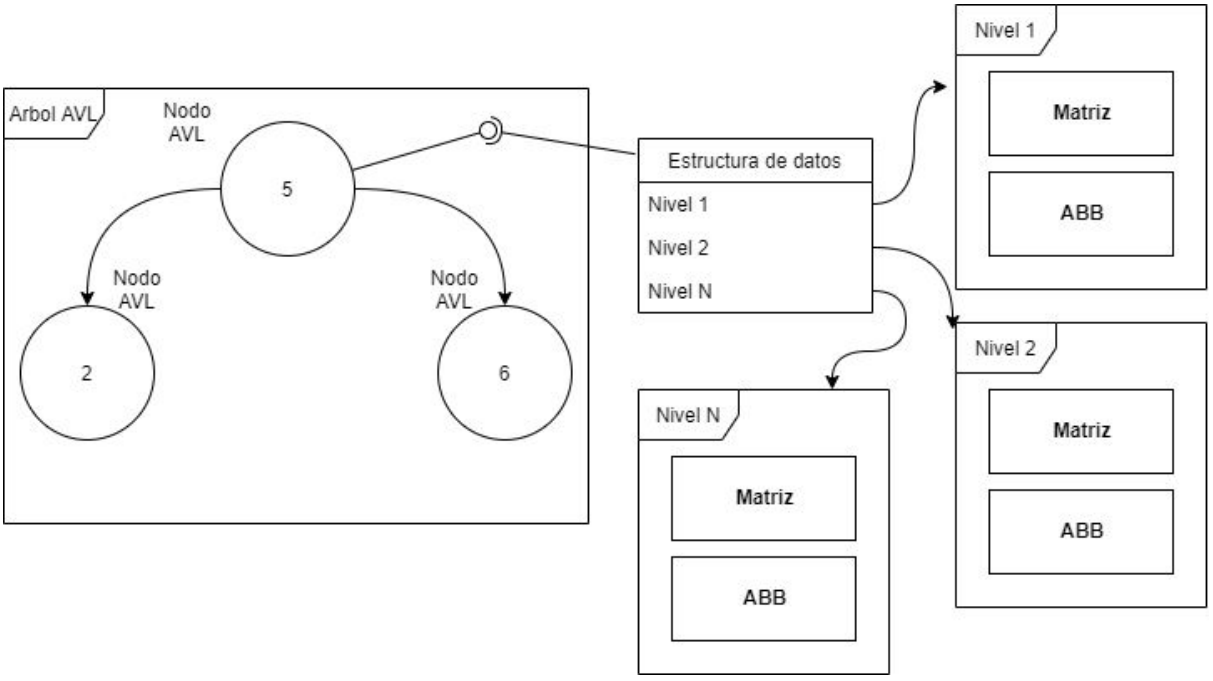
ABB: cuando usamos este término nos referimos a un árbol binario de búsqueda.

Matriz dispersa: Es una estructura de datos no lineal.

Objeto: Cuando hacemos referencia a un objeto es por ejemplo un televisor, un mueble, una mesa, este adentro debe tener su id, descripción, color, un arreglo de coordenadas donde están posicionados siempre referentes a su nodo superior izquierdo por ejemplo **(0,0),(0,1),(1,0),(1,1)** donde su nodo superior izquierdo es 0,0 a esto es donde le sumaremos la posición cuando lo queramos insertar por ejemplo si se desea insertar en la **posición (1,1)** de la matriz dispersa entonces sus sus coordenadas serán **(1,1)(1,2)(2,1)(2,2)**

JSON: Notación de Objetos de JavaScript

Vista General del proyecto



Consideraciones

- El lenguaje a utilizar será **C++**.
- IDE a utilizar libre
- La entrega será por medio de la plataforma de **UEDI**.
- Todas las estructuras de datos deben ser desarrolladas por el estudiante.
- El estudiante debe tener un repositorio privado en **github** y agregar a su tutor como contribuidor al repositorio del proyecto (**Cada tutor les hará llegar su usuario**)
- Será calificado del último commit realizado antes de la fecha de entrega.
- Entregables:
    - Código Fuente (Sin dependencias rotas).
- **Fecha de entrega: 17 de Septiembre de 2020 antes de las 23:59 horas.**
- Copias totales o parciales serán penalizadas con nota de 0 puntos y reportadas ante escuela de ciencias y sistemas. **(Se utilizará una herramienta para validar copias entre alumnos de una misma sección y entre secciones).**
- **Todos los reportes deben estar generados con graphviz.**
- **Todo tendrá un enfoque a programación orientada a objetos, por lo tanto se revisará que los estudiantes utilicen clases, si en dado caso se utilizan structs el apartado donde se utilizó tendrá 0.**