UNIVERSIDAD CATÓLICA BOLIVIANA “SAN PABLO”

CARRERA DE INGENIERÍA INDUSTRIAL

SEDE SANTA CRUZ

INFORME DEL JUEGO SOKOBAN

Integrantes:

- Giovani Hugo Morales Riglos
- Leonardo David Rivero López

Docente: Ing. Eddy Escalante Ustariz

Fecha de presentación: 12/12/2025.

Santa Cruz - Bolivia

1. Introducción

Este proyecto busca aplicar de forma práctica los conocimientos aprendidos sobre matrices, utilizando su lógica de posiciones para desarrollar un juego basado en la mecánica de SOKOBAN. Cada número dentro de la matriz representa un elemento del escenario como el avatar, las cajas, las minas y los espacios libres obteniendo así mismo varios niveles interactivos con distintas posiciones.

De esta manera, cada movimiento realizado por el avatar se refleja en la matriz mediante cambios de posición. Con esto demostramos que los conceptos de matrices no solo sirven para ejercicios matemáticos, sino también para controlar mapas, mover objetos y manejar la lógica interna de un videojuego sencillo.
Además, la estructura del juego permite visualizar de forma directa cómo una modificación en la matriz impacta en el entorno del jugador. Esto hace que el aprendizaje sea más intuitivo. El enfoque elegido facilita entender la importancia de las posiciones y los desplazamientos dentro de una matriz.

- **Importancia del proyecto**

Este proyecto es importante porque permite comprender de manera clara cómo se utilizan las matrices en programación. El juego muestra que acciones como mover una caja, avanzar en el mapa o evitar una mina son en realidad modificaciones en los índices de la matriz. Así, el jugador aprende mientras juega, entendiendo que la estructura del mapa depende directamente de la organización de los datos dentro de una matriz. Gracias a esto, el proyecto combina aprendizaje con entretenimiento, facilitando que estudiantes como nosotros entiendan cómo un concepto matemático puede convertirse en la base funcional de un videojuego de manera que el m estudiantes pueda interactuar ya se modificando el cómo jugándolo a su manera. También ayuda a reforzar la lógica de decisión, ya que el jugador debe analizar posiciones permitidas y obstáculos.

Esto demuestra que la programación y las matemáticas pueden integrarse en actividades prácticas y creativas.

3. Desarrollo del Proyecto

El proyecto se basa en aplicar los conocimientos adquiridos sobre matrices para crear un sistema que procesa datos mediante posiciones organizadas, permitiendo automatizar cálculos y mejorar la lógica del programa. Para ello se definieron matrices que almacenan información, se programaron operaciones básicas como suma, resta y búsqueda de valores, y se estructuró un flujo que interpreta cada posición como parte de una solución. A lo largo del desarrollo se depuró el código para optimizar el uso de memoria y reducir errores. Finalmente, se realizaron pruebas con diferentes conjuntos de datos para verificar la funcionalidad y precisión del sistema.

3.1. Diseño del Videojuego

● Descripción del concepto del videojuego

En un antiguo almacén perdido entre montañas, tres exploradores fueron llamados para resolver un misterio: las cajas mágicas que mantienen el equilibrio del lugar comenzaron a desordenarse solas. Cada avatar —un ingeniero, una aventurera y un robot ayudante— debe recorrer distintos niveles del almacén para empujar las cajas a sus pedestales energéticos. A medida que avanzan los tres niveles, los pasillos se vuelven más estrechos y los desafíos más complejos. Solo organizando todas las cajas en su sitio podrán restaurar la energía del almacén y evitar que colapse para siempre.

● Mecánicas principales del juego:

¿Qué acciones puede realizar el jugador?

- Escoger cualquiera de los 3 niveles para jugar
- Escoger cualquiera de los 3 tipos de avatar
- Desplazarse a lo largo del mapa
- Mover objetos para tapar las minas en un tiempo determinado

○ ¿Qué condiciones determinan ganar o perder?

● Descripción de los niveles:

cambia de dificultad en el desplazamiento del mapa y tiene menos tiempo para lograr tapar las minas explosivas

3.2. Herramientas Utilizadas

Explica brevemente las tecnologías usadas:

● Lenguaje de programación (JavaScript).

● Software (Visual Studio Code).

● Repositorio de GitHub

3.3. Integración con el Dobot MG400

● Pasos realizados para configurar el Dobot con DobotStudio Pro.

1.-primero conectamos el dobot a la computadora con el cable ethernet, conectar la fuente de poder al dobot , instalar dobot studio pro en el dobot y conectar el botón de emergencia, conectar a la aplicacion

● Explicación de cómo el Dobot interactúa con el videojuego: ○ ¿Qué acciones controla?

○ ¿Cómo afecta la jugabilidad?

Afecta en el retraso entre presionar teclas , y al mismo tiempo pierde tiempo y agilidad del pesonaje

3.4. Organización del Trabajo

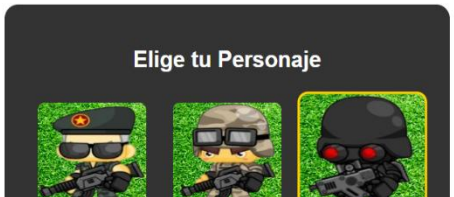
● Explica cómo se dividieron las tareas en el equipo (¿quién hizo qué?).

Mi persona se encargo de hacer tanto el informe con las diapositivas y recolección de las imágenes que se ocuparon en el juego

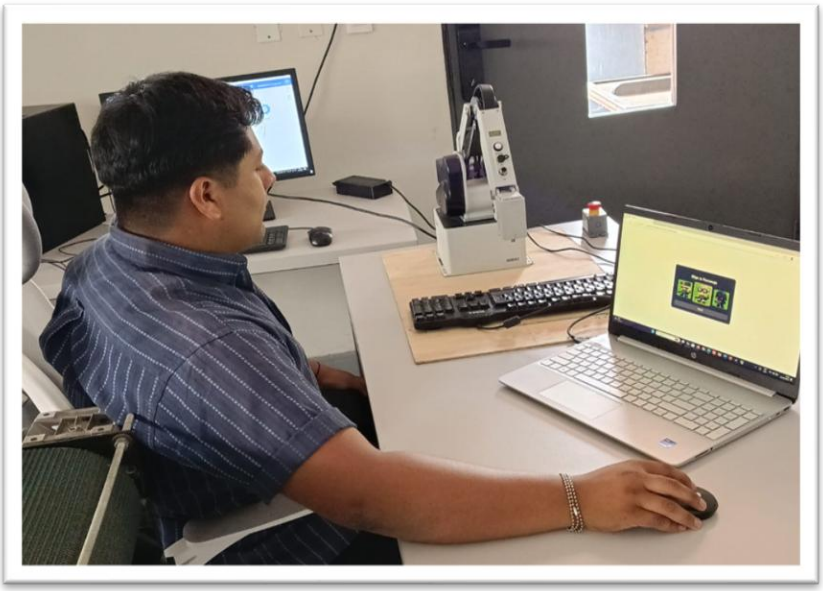
● Uso del repositorio en GitHub indicando que aporte realizó cada integrante.

4. Resultados

● Capturas de pantalla del videojuego funcionando.



● Fotografía de interacción del botón con el juego



5. Conclusiones y Lecciones Aprendidas

● Reflexión sobre:

○ Retos enfrentados durante el desarrollo.

Al principio de la elaboración del código fue hacer que el avatar del juego se moviera ya que luego de algunos intentos modificando el código recién se pudo hacer que el avatar se moviera

○ Soluciones implementadas.

La solución por parte de mi compañero fue modificar el código para que el avatar se lograra mover por el mapa

○ Habilidades adquiridas.

1. Aprender a implementar un Chat Bot interactivo con el jugador en el juego
2. Aprender hacer un juego con dinámico con sonidos, movimiento
3. Trabajar en Equipo

● Recomendaciones para mejorar el proyecto.

Se podría añadir más niveles en el juego que fueran más dinámicos y con mas dificultades añadiendo avatares que si tocan o disparan al avatar del jugador perdiera automáticamente.

6. Anexos

● Fragmentos de código importantes.

- **<u>FRAGMENTOS DE LA CLASE GAME</u>**

FRAGMENTO 1 — Inicialización del juego y carga del mapa

Este fragmento es clave porque muestra cómo se configura el canvas, la matriz y el jugador.

```
constructor(canvasId, mapMatrix, images) {
  this.canvas = document.getElementById(canvasId);
  this.ctx = this.canvas.getContext("2d");

  this.rows = mapMatrix.rows || 10;
  this.cols = mapMatrix.cols || 10;
  this.cellSize = this.canvas.width / this.cols;

  this.mapMatrix = mapMatrix;
  this.images = images;

  this.player = this.initPlayer();
  this.initControls();
}
```

FRAGMENTO 2 — Sistema de temporizador del nivel

Es importante porque muestra la mecánica de presión/tiempo límite.

```
 startTimer() {
   if (timerActive) return;

   timerActive = true;

   timerInterval = setInterval(() => {
     timeLeft--;

     if (timeLeft <= 0) {
       timeLeft = 0;
       clearInterval(timerInterval);
       showGameOver();
     }

     this.draw();
   }, 1000);
 }
```

FRAGMENTO 3 — Detección e inicialización del avatar seleccionado

Fragmento esencial porque conecta tu selección de personaje con el mapa.

```
initPlayer() {
  const selectedCharacter = parseInt(localStorage.getItem("selectedCharacter") ||
  const validPlayers = [8, 9, 10];

  for (let r = 0; r < this.rows; r++) {
    for (let c = 0; c < this.cols; c++) {
      if (validPlayers.includes(this.mapMatrix.getValue(r, c))) {
        return new Player(r, c, this.mapMatrix.getValue(r, c));
      }
    }
  }

  return new Player(0, 0, selectedCharacter);
}
```

FRAGMENTO 4 — Controles del jugador (teclas y movimientos)

Fundamental porque muestra cómo se maneja la interacción del usuario.

```
initControls() {
  document.addEventListener("keydown", (e) => {
    switch (e.key) {
      case "ArrowUp":
        this.handleMove(-1, 0);
        new Audio("assets/sounds/move.mp3").play();
        break;

      case "ArrowDown":
        this.handleMove(1, 0);
        new Audio("assets/sounds/move.mp3").play();
        break;

      case "ArrowLeft":
        this.handleMove(0, -1);
        new Audio("assets/sounds/move.mp3").play();
        break;

      case "ArrowRight":
        this.handleMove(0, 1);
        new Audio("assets/sounds/move.mp3").play();
        break;
    }
  });
}
```

FRAGMENTO 5 — Lógica de movimiento, puntaje y limpieza del inicio

Es el más importante del juego porque contiene la lógica central.

```
handleMove(dr, dc) {
  this.startTimer();

  const beforeRow = this.player.row;
  const beforeCol = this.player.col;

  this.player.move(dr, dc, this.rows, this.cols, this.mapMatrix);

  const moved = this.player.row !== beforeRow || this.player.col !== beforeCol;

  if (moved) {
    score -= 10;

    if (!this.startCellCleared) {
      this.mapMatrix.setValue(beforeRow, beforeCol, 1);
      this.startCellCleared = true;
    }
  }

  this.draw();
}
```

FRAGMENTO 6 — Dibujado del juego (render del mapa y personaje)

Clave para demostrar cómo se renderiza cada frame del juego.

```
draw() {
this.ctx.clearRect(0, 0, this.canvas.width, this.canvas.height);

  for (let r = 0; r < this.rows; r++) {
    for (let c = 0; c < this.cols; c++) {
      const value = this.mapMatrix.getValue(r, c);
      const img = this.images[value];
      if (img) {
        this.ctx.drawImage(
          img,
          c * this.cellSize,
          r * this.cellSize,
          this.cellSize,
          this.cellSize
        );
      }

      if (r === this.player.row && c === this.player.col) {
        const playerImg = this.images[this.player.value];
        if (playerImg) {
          this.ctx.drawImage(
            playerImg,
            c * this.cellSize,
            r * this.cellSize,
            this.cellSize,
            this.cellSize
          );
        }
      }
    }
  }

  drawHUD(this.ctx, this.canvas);
}
```

FRAGMENTO 7 — HUD (tiempo y score en pantalla)

Demuestra interfaz gráfica básica.

```
function drawHUD(ctx, canvas) {
  ctx.fillStyle = "rgba(0,0,0,0.6)";
  ctx.fillRect(0, 0, canvas.width, 40);

  ctx.fillStyle = "white";
  ctx.font = "20px Arial";
  ctx.fillText("Tiempo: " + timeLeft, 20, 25);
  ctx.fillText("Score: " + score, 200, 25);
}
```

- **FRAGMENTOS DE LA CLASE PALYER**

FRAGMENTO 1 — Constructor del jugador

Esencial porque define la posición inicial y el tipo de avatar.

```
constructor(row, col, value = 10) {
  this.row = row;
  this.col = col;
  this.value = value;
}
```

FRAGMENTO 2 — Cálculo de la nueva posición

Fragmento clave: determina hacia dónde quiere moverse el jugador.

```
move(dr, dc, rows, cols) {
  const newRow = this.row + dr;
  const newCol = this.col + dc;

  // Evitar salirse del mapa
  if (newRow < 0 || newRow >= rows) return;
  if (newCol < 0 || newCol >= cols) return;

  const cellValue = mapMatrix.getValue(newRow, newCol);
```

FRAGMENTO 3 — Bloqueos (muros, agua, ladrillos)

Este es importante porque controla qué celdas NO se pueden atravesar.

```
// Bloqueo: muros, ladrillos, agua (según tu lógica)
if (cellValue === 1 || cellValue === 3 || cellValue === 4) return;
```

FRAGMENTO 4 — Sistema de empuje de bloques

*Este es el fragmento **más importante** porque representa la mecánica principal estilo Sokoban.*

```
// Lógica para empujar ladrillos
const targetCell = mapMatrix.getValue(newRow, newCol);
if (targetCell === 6) {
  const nextRow = newRow + dr;
  const nextCol = newCol + dc;

  // Verificar que la siguiente celda esté vacía o transitable
  if (!mapMatrix.isValidPosition(nextRow, nextCol)) return;

  const nextCell = mapMatrix.getValue(nextRow, nextCol);
  if (nextCell === 0 || nextCell === 2) {
    mapMatrix.setValue(nextRow, nextCol, 6); // mover el bloque
    mapMatrix.setValue(newRow, newCol, 2);   // dejar la celda anterior libre
  } else {
    return; // No se puede empujar
  }
}
```

FRAGMENTO 5 — Actualización final de la posición

Indica que el movimiento fue exitoso.

```
this.row = newRow;
this.col = newCol;
}
```

Rúbrica de Evaluación Criterio	Descripción	Puntaje Máximo
Funcionalidad del videojuego	El videojuego funciona correctamente, incluye 2 niveles y es jugable con mecánicas básicas claras.	10 puntos
Integración con el Dobot	Uso eficiente del Dobot MG400 como controlador, con implementación de funciones específicas.	20puntos
Informe	Cumple con los apartados detallados, tiene un análisis claro y presenta conclusiones relevantes.	10 puntos
Presentación y defensa del proyecto	Demostración efectiva del videojuego y modificación solicitada por el docente.	60 puntos