ESTRUCTURA DE DATOS

TAKEN CON DEGRADADO RGB

Guarderas David

Guevara Mathías

Tinoco Shared

Torres Sebastián

2022

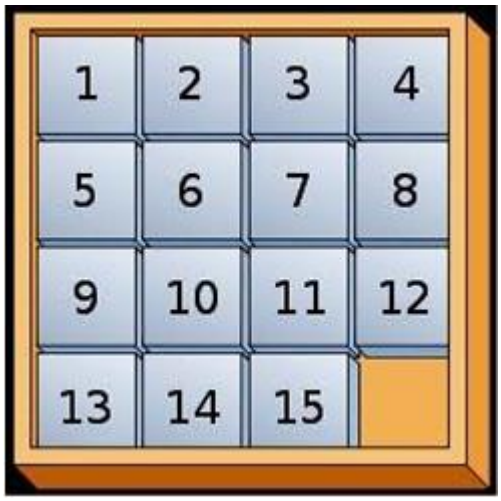
1. INTRODUCCIÓN

El objetivo de este proyecto es crear un programa para el juego de Taken de orden (n x n) con colores para poder visualizar un efecto de degradado, dicho n es elegido por el usuario. El programa brinda al usuario dos opciones para resolverlo, la primera donde el usuario puede mover las fichas con las flechas de dirección, y otra donde se define un algoritmo capaz de resolver el juego por sí solo.

2. EL PUZZLE

a. Definición

El puzzle de (n^2 - 1) consiste en un tablero de (n x n) con n casillas, numeradas del 1 al (n^2 - 1), ordenadas de manera aleatoria. La casilla n^2 se encuentra vacía para poder realizar los movimientos necesarios, de forma horizontal o vertical, para poder cambiar el orden de las casillas. Recibe multitud de nombres según el número de fichas 8-puzzle para la versión 3x3, 15-puzzle para la versión de 4x4, u otros nombres menos comunes como Puzzle Gem, Puzzle Boss, Cuadrado Místico o Taken. El objetivo del rompecabezas es ordenar las fichas utilizando el espacio vacío para desplazarlas.



b. Historia

El rompecabezas deslizante más antiguo que se conoce es el Juego del 15, inventado por Noyes Chapman en 1880. A menudo se atribuye erróneamente a Sam Loyd el hacer popular este rompecabezas con su versión imposible del Juego del 15, pero fue Chapman quien desató la locura a principios de dicha década.

De los años 50 a los 80 los rompecabezas deslizantes basados en letras para formar palabras fueron muy populares. Este tipo de rompecabezas tienen varias soluciones, como puede verse en ejemplos como Ro-Let (un juego del 15 basado en letras), Scribe-o (4x8), y Lingo.

El Juego de 15 ha sido digitalizado (como videojuego de rompecabezas) y hay ejemplos disponibles en línea que permiten jugar gratuitamente. Algunas versiones se parecen a los puzles jigsaw en que el objetivo es formar una imagen en la pantalla, pero en general el objetivo es siempre ordenar según números en cada pieza

3. FUNCIONES

3.1. Crear Paleta

Se inicializa un tablero del juego de tamaño (n x n) cuyo orden es indicado por el usuario.

```
void Paleta::crearPaleta(int excluido)
{


	int xaux=xIn,yaux=yIn;
	for(int i=0;i<dim;i++)
	{
		for(int j=0;j<dim;j++)
		{
			(*(*(paleta+i)+j)).setX(xaux);
			(*(*(paleta+i)+j)).setY(yaux+300);
			(*(*(desorden+i)+j)).setX(xaux);
			(*(*(desorden+i)+j)).setY(yaux);
			(*(*(paletita+i)+j)).setX(xaux);
			(*(*(paletita+i)+j)).setY(yaux);
			(*(*(paleta+i)+j)).setRad(20);
			(*(*(paletita+i)+j)).setRad(20);
			(*(*(desorden+i)+j)).setRad(20);
			yaux+=40;
		}
		xaux+=40;
		yaux=yIn;

	}
}
```

3.2. Degradar

Para el proyecto el tablero Taken no es como el convencional (lleno con números) en su lugar son colores con un efecto degradado, entre los colores primarios. (rojo-azul) (verde-rojo) (azul-verde).

Una vez que se haya indicado desde qué color iniciar, se calcula los RGB de cada cuadro en los bordes superior e izquierdo, de manera que vayan en degradado.

Ya llenos estos bordes se llena la siguiente fila hasta la fila n, tomando como valor inicial el primer cuadro de la fila.

```
void Paleta::degradar(int col)
{
	int roj=0,ver=0,azu=0,rojc=0,verc=0,azuc=0;
	int rojaux=0,veraux=0,azuaux=0;
	if(col==1)
	{
		roj=255;
		azuc=255;
	}
	else if(col==2)
	{
		ver=255;
		rojc=255;
	}
	else if(col==3)
	{
		azu=255;
		verc=255;
	}

	(*(*(paleta))).setColor(roj,ver,azu);

	rojaux=roj;
	azuaux=azu;
	veraux=ver;

	for(int i=1;i<dim;i++)
	{
		if(i!=dim-1)
		{
			if(col==1)
			{
				rojaux-=(255/(2*(dim-1)));
				azuaux+=(255/(2*(dim-1)));
			}
			else if(col==2)
			{
				veraux-=(255/(2*(dim-1)));
				rojaux+=(255/(2*(dim-1)));
			}
			else if(col==3)
			{
				azuaux-=(255/(2*(dim-1)));
				veraux+=(255/(2*(dim-1)));
```

```
                }
        }
        else
        {
                rojaux=roj;
                azuaux=azu;
                veraux=ver;
                if(col==1)
                {
                        rojaux-=(255/(2));
                        azuaux+=(255/(2));
                }
                else if(col==2)
                {
                        veraux-=(255/(2));
                        rojaux+=(255/(2));
                }
                else if(col==3)
                {
                        azuaux-=(255/(2));
                        veraux+=(255/(2));
                }
        }


        (*(*(paleta)+i)).setColor(rojaux,veraux,azuaux);
        (*(*(paleta+i))).setColor(rojaux,veraux,azuaux);
}

rojaux=roj;
azuaux=azu;
veraux=ver;

for(int j=1;j<dim;j++)
{
        rojaux=(*(*(paleta+j))).getR();
        azuaux=(*(*(paleta+j))).getB();
        veraux=(*(*(paleta+j))).getG();
        for(int i=1;i<dim;i++)
        {
                if(col==1)
                {
                        rojaux-=(255/(2*(dim-1)));
                        azuaux+=(255/(2*(dim-1)));
                }
                else if(col==2)
                {
```

```
                                veraux-=(255/(2*(dim-1)));
                                rojaux+=(255/(2*(dim-1)));
                        }
                        else if(col==3)
                        {
                                azuaux-=(255/(2*(dim-1)));
                                veraux+=(255/(2*(dim-1)));
                        }

                        (*(*(paleta+i)+j)).setColor(rojaux,veraux,azuaux);

                }
        }

        (*(*(paleta+dim-1)+dim-1)).setColor(255,255,255);
}
```

3.3. Mostrar Paleta Completa

Se imprime el tablero por consola con el modo gráfico de la librería en c++ con el color de cada cuadro.

```
void Paleta::mostrarCompleto()
{
        for(int i=0;i<dim;i++)
        {
                for(int j=0;j<dim;j++)
                {
                        (*(*(paleta+j)+i)).mostrar();
                }
        }
}
```

3.4. Jugar

Con el tablero desordenado, el programa reconoce las teclas arriba, abajo, izquierda, derecha, enter y escape para los comandos básicos de juego. Dependiendo de la tecla de dirección que presione el usuario la pieza adyacente al cuadro vacío intercambiará de lugar con este. Las teclas enter y escape sirven para comprobar si el juego fue realizado correctamente o salir a las opciones para elegir un color respectivamente. Incluye también la tecla “r” para **r**efrescar el cuadro de juego función añadida para una mejor experiencia gráfica dado que muy rara vez algunos pixeles no se observan del todo, con esta función se solucionó ese problema. Las teclas “t”y “p” generan un txt y un pdf. Y por último las teclas “b” y “c” enfocados en realizar un **b**ackup de la matriz con los códigos RGB del cuadro actual y **c**argar dicho archivo para continuar el juego desde ese punto.

```
void Paleta::jugar()
```

```
{
	int xa,ya;
	int tecla;
	mostrarCompleto();
	ocultarCursor();
	while(tecla!=ESCAPE)
	{
		mostrarPaleta();
		xa=xW;
		ya=yW;
		do
		{
			tecla=getch();
		}while(tecla!=TECLA_ARRIBA && tecla!=TECLA_ABAJO &&
tecla!=TECLA_IZQUIERDA && tecla!=TECLA_DERECHA
		&& tecla!=ENTER && tecla !=ESCAPE && tecla !=T && tecla !=B &&
tecla !=P && tecla !=R && tecla !=C);

		switch(tecla)
		{
			case TECLA_ARRIBA:
			{
				if(xW+1<dim)
				{
					movimientos++;
					xW++;
				}

				break;
			}
			case TECLA_ABAJO:
			{
				if(xW-1>=0)
				{
					movimientos++;
					xW--;
				}
				break;
			}
			case TECLA_DERECHA:
			{
				if(yW-1>=0)
				{
					movimientos++;
					yW--;
				}
				break;
```

```
                    }
                    case TECLA_IZQUIERDA:
                    {
                            if(yW+1<dim)
                            {
                                    movimientos++;
                                    yW++;
                            }
                            break;
                    }
                    case ENTER:
                    {
                            if(comprobar())
                            {
                                    gotoxy(62,30);
                                    std::cout<<"\t\tFelicidades!";
                                    gotoxy(62,31);
                                    std::cout<<"\t\tHa completado el puzzle
exitosamente.";
                                    gotoxy(69,32);
                                    system("pause");
                                    system("cls");
                                    mostrarCompleto();
                            }else{
                                    gotoxy(62,30);
                                    std::cout<<"\t\tLo sentimos!";
                                    gotoxy(62,31);
                                    std::cout<<"\t\tNo ha completado el puzzle
exitosamente.";
                                    gotoxy(69,32);
                                    system("pause");
                                    system("cls");
                                    mostrarCompleto();
                            }
                            break;
                    }
                    case T:
                    {
                            generarArchivosTxT();
                            gotoxy(62,32);
                            system("pause");
                            system("cls");
                            mostrarCompleto();
                            break;
                    }
                    case P:
                    {
```

```
                        generarArchivosPDF("RGB.pdf");
                        gotoxy(62,32);
                        system("pause");
                        system("cls");
                        mostrarCompleto();
                        break;
                }
                case B:
                {
                        generarBackup();
                        gotoxy(62,32);
                        system("pause");
                        system("cls");
                        mostrarCompleto();
                        break;
                }
                case C:
                {
                        leerBackup();
                        xW=xNueva;
                        xa=xNueva;
                        yW=yNueva;
                        ya=yNueva;
                        gotoxy(62,32);
                        system("pause");
                        system("cls");
                        mostrarCompleto();
                        break;
                }
                case R:
                {
                        system("cls");
                        mostrarCompleto();
                        break;
                }
        }

        (*(*(desorden+ya)+xa)).setR((*(*(desorden+yW)+xW)).getR());
        (*(*(desorden+ya)+xa)).setG((*(*(desorden+yW)+xW)).getG());
        (*(*(desorden+ya)+xa)).setB((*(*(desorden+yW)+xW)).getB());

        (*(*(desorden+yW)+xW)).setColor(255,255,255);
    }
}
```

3.5. Generar Archivo txt

El programa permite generar un archivo .txt de la códigos RGB de las posiciones respectivas del cuadrado actual de juego.

```
void Paleta::generarArchivosTxT()
{
	ofstream archivo;
	string dirS;
	gotoxy(62,30);
	cout<<"Ingrese la direccion en la que desea que se genere el archivo: ";
	getline(cin,dirS);
	char *dir=new char[1000];
	memmove(dir,dirS.c_str(),dirS.length());
	int aux2=dirS.length()+1,j=0;
	char aux1[1000];
	for(int i=0;i<=aux2;i++)
	{
		if(dir[i]=='\\')
		{
			for(int k=i+1;k<=aux2;k++)
			{
				aux1[j]=dir[k];
				j++;
			}
			dir[i]='/';
			dir[i+1]='/';
			aux2++;
			j=0;
			for(int l=i+2;l<=aux2;l++)
			{
				dir[l]=aux1[j];
				j++;
			}
			j=0;
		}
	}
	string dirSR(dir);
	dirSR+="//matriz.txt";
	escribirArchivo(archivo,dirSR);
	gotoxy(62,31);
	cout<<"Archivo de texto generado con exito en la direccion: "<<dirSR;
}
```

3.6. Generar Archivo PDF

El programa permite generar un archivo .pdf de la códigos RGB de las posiciones respectivas del cuadrado actual de juego.

```
void Paleta::generarArchivosPDF(string nombre)
{
        PDF p;
   p.setFont(PDF::HELVETICA_BOLD,30);
        p.showTextXY("TAKEN",230,746);
        p.setFont(PDF::HELVETICA,10);
        p.showTextXY("Grupo 5",50,699);
        p.showTextXY("David Guarderas, Mathias Guevara",50,679);
        p.showTextXY("Shared Tinoco, Sebastian Torres",50,659);
   p.showTextXY("Datos RGB:",50,546);
   string linea, texto;
   int y=516;
   int x=50;
   ifstream original("matriz.txt");
   while (getline(original, linea))
   {
      p.showTextXY(linea,x,y);
      y-=15;
   }
   original.close();
        string errMsg;
         if (!p.writeToFile(nombre, errMsg))
   {
      cout << errMsg << endl;
   }
   else
   {
      cout << "(File Successfully Written)" << endl;
   }

   cout << endl;

        gotoxy(62,31);
   cout<<"Archivo PDF generado correctamente";
   cout << endl;
}
```

3.7. Generar Archivo PDF

El programa permite generar un archivo .pdf de la códigos RGB de las posiciones respectivas del cuadrado actual de juego.

```
void Paleta::generarArchivosPDF(string nombre)
{
        PDF p;
   p.setFont(PDF::HELVETICA_BOLD,30);
        p.showTextXY("TAKEN",230,746);
```

```
        p.setFont(PDF::HELVETICA,10);
        p.showTextXY("Grupo 5",50,699);
        p.showTextXY("David Guarderas, Mathias Guevara",50,679);
        p.showTextXY("Shared Tinoco, Sebastian Torres",50,659);
   p.showTextXY("Datos RGB:",50,546);
   string linea, texto;
   int y=516;
   int x=50;
   ifstream original("matriz.txt");
   while (getline(original, linea))
   {
      p.showTextXY(linea,x,y);
      y-=15;
   }
   original.close();
        string errMsg;
         if (!p.writeToFile(nombre, errMsg))
   {
      cout << errMsg << endl;
   }
   else
   {
      cout << "(File Successfully Written)" << endl;
   }

   cout << endl;

        gotoxy(62,31);
   cout<<"Archivo PDF generado correctamente";
   cout << endl;
}
```

3.8. Generar Backup

El programa permite generar un backup de los archivos txt generados hasta el momento actual.

```
void Paleta::generarBackup()
{
        time_t t;
   t=time(NULL);
   struct tm *f;
   f=localtime(&t);

   int dd,mm,aaaa,hora,min;
   dd=f->tm_mday;
   mm=f->tm_mon+1;
```

```
   aaaa=f->tm_year+1900;
   hora=f->tm_hour;
   min=f->tm_min;

   string linea;
   string txt;
   string fecha;
   string aaaa1=(to_string(aaaa));
   string mm1=(to_string(mm));
   string dd1=(to_string(dd));
   string h1=(to_string(hora));
   string m1=(to_string(min));
        fecha                =                "Respaldos\\"+dd1+"-"+mm1+"-"+aaaa1+"-
"+h1+"h"+m1+"m"+"Backup"+".txt";

   ofstream backup;
   backup.open(fecha);
        for(int i=0;i<dim;i++)
        {
                for(int j=0;j<dim;j++)
                {
                        backup<<(*(*(desorden+j)+i)).getR();
                        backup<<"\n";
                        backup<<(*(*(desorden+j)+i)).getG();
                        backup<<"\n";
                        backup<<(*(*(desorden+j)+i)).getB();
                        backup<<"\n";
                }
        }

        for(int i=0;i<dim;i++)
        {
                for(int j=0;j<dim;j++)
                {
                        backup<<(*(*(paleta+j)+i)).getR();
                        backup<<"\n";
                        backup<<(*(*(paleta+j)+i)).getG();
                        backup<<"\n";
                        backup<<(*(*(paleta+j)+i)).getB();
                        backup<<"\n";
                }
        }
   backup.close();
   gotoxy(62,29);
        cout<<"Archivo de texto generado con exito en la direccion: "<<fecha;
}
```

3.9. Cargar Backup

El programa permite leer los backups generados para poder leer los últimos datos del juego cuando se generó el archivo, y así poder continuar desde ese punto.

```
void Paleta::leerBackup()
{
	ofstream archivo;
	string dirS;
	gotoxy(62,30);
	cout<<"Ingrese la direccion en la que desea que se genere el archivo: ";
	getline(cin,dirS);
	char *dir=new char[500];
	memmove(dir,dirS.c_str(),dirS.length());
	int aux2=dirS.length()+1,j=0;
	char aux1[500];
	for(int i=0;i<=aux2;i++)
	{
		if(dir[i]=='\\')
		{
			for(int k=i+1;k<=aux2;k++)
			{
				aux1[j]=dir[k];
				j++;
			}
			dir[i]='/';
			dir[i+1]='/';
			aux2++;
			j=0;
			for(int l=i+2;l<=aux2;l++)
			{
				dir[l]=aux1[j];
				j++;
			}
			j=0;
		}
	}
	string dirSR(dir),linea;
	int o=0,apNum=0,roj,azu,ver,k=0,l=0,idk=0;

	ifstream archivo1(dirSR.c_str());
	gotoxy(0,31);
	while(getline(archivo1, linea))
	{
		char *cline=new char[10];
		memmove(cline,linea.c_str(),linea.length());
		if(o==0)
		{
```

```
                        roj=atoi(cline);
                        o++;
                }else if(o==1)
                {
                        ver=atoi(cline);
                        o++;
                }else if(o==2)
                {
                        o=0;
                        azu=atoi(cline);
                        if(apNum<dim*dim)
                        {
                                (*(*(desorden+l)+k)).setColor(roj,ver,azu);
                                if(roj==255 && ver==255 && azu==255)
                                {
                                        xNueva=l;
                                        yNueva=k;
                                }
                        }else
                        {
                                (*(*(paleta+l)+k)).setColor(roj,ver,azu);
                        }
                        if(l<dim-1)
                        {
                                l++;
                        }else
                        {
                                l=0;
                                if(k<dim-1)
                                {
                                        k++;
                                }else
                                {
                                        l=0;
                                        k=0;
                                }
                        }
                        apNum++;
                }
        }
}
```

3.9. Comprobar

Esto nos permite comprobar el cuadrado actual con el cuadrado final para poder comprobar que este se haya realizado correctamente

```
bool Paleta::comprobar()
{
	for(int i=0;i<dim;i++)
	{
		for(int j=0;j<dim;j++)
		{
			if(((*(*(desorden+i)+j)).getR()!=(*(*(paleta+i)+j)).getR()) &&
((*(*(desorden+i)+j)).getG()!=(*(*(paleta+i)+j)).getG()) &&
((*(*(desorden+i)+j)).getB()!=(*(*(paleta+i)+j)).getB()))
			{
				return false;
				break;
			}
		}
	}
	return true;
}
```

3.10. Solucionar

Este método permite solucionar automáticamente un cuadrado generado por el programa.

```
void Paleta::solucionar()
{
	int x=0,y=0;
	for(int i=0;i<dim;i++)
	{
		for(int j=0;j<dim;j++)
		{
			if(((*(*(desorden+j)+i)).getR()==(*(*(paleta+y)+x)).getR()) &&
((*(*(desorden+j)+i)).getG()==(*(*(paleta+y)+x)).getG()) &&
((*(*(desorden+j)+i)).getB()==(*(*(paleta+y)+x)).getB()))
			{
				xB=i;
				yB=j;
				xL=x;
				yL=y;

				if(y<dim-1)
				{
					y++;
				}
				else
				{
```

```
                                        y=0;
                                        if(x<dim-1)
                                        {
                                                x++;
                                        }else
                                        {
                                                x=0;
                                                y=0;
                                        }
                                }
                                break;
                        }
                }
        }
        //mover();
        if(comprobar())
        {
                return;
        }

}
```

3.11. Mover

```
void Paleta::mover()
{
        bool encima=false,derecha=false;
        int xa,ya;
        if(xB>=1)
        {
                while(!encima)
                {
                        mostrarPaleta();
                        xa=xW;
                        ya=yW;
                        if(xB==xL)
                        {
                                encima=true;
                        }else if(xW==xB-1)
                        {
                                if(yW==yB)
                                {
                                        xW++;
                                        xB--;
                                }
```

```
                        else if(yW>yB)
                        {
                                yW--;
                        }else
                        {
                                yW++;
                        }
                }else if(xW==xB)
                {
                        xW--;
                }else if(xW>xB)
                {
                        if(yW==yB)
                        {
                                if(yW==dim-1)
                                {
                                        yW--;
                                }else{
                                        yW++;
                                }
                        }else
                        {
                                xW--;
                        }
                }else{
                        xW++;
                }
                (*(*(desorden+ya)+xa)).setR((*(*(desorden+yW)+xW)).getR());
                (*(*(desorden+ya)+xa)).setG((*(*(desorden+yW)+xW)).getG());
                (*(*(desorden+ya)+xa)).setB((*(*(desorden+yW)+xW)).getB());

                (*(*(desorden+yW)+xW)).setColor(255,255,255);
                cin.ignore();
        }
}
if(yB>=1)
{
        while(!derecha)
        {
                mostrarPaleta();
                xa=xW;
                ya=yW;
                if(yB==yL)
                {
                        derecha=true;
                }else if(yW==yB-1)
                {
```

```
                                if(xW==xB)
                                {
                                        yW++;
                                        yB--;
                                }
                                else if(xW>xB)
                                {
                                        xW--;
                                }else
                                {
                                        xW++;
                                }
                        }else if(yW==yB)
                        {
                                yW--;
                        }else if(yW>yB)
                        {
                                if(xW==xB)
                                {
                                        if(xW==0)
                                        {
                                                xW++;
                                        }else{
                                                xW--;
                                        }
                                }else
                                {
                                        yW--;
                                }
                        }else{
                                yW++;
                        }
                        (*(*(desorden+ya)+xa)).setR((*(*(desorden+yW)+xW)).getR());
                        (*(*(desorden+ya)+xa)).setG((*(*(desorden+yW)+xW)).getG());
                        (*(*(desorden+ya)+xa)).setB((*(*(desorden+yW)+xW)).getB());

                        (*(*(desorden+yW)+xW)).setColor(255,255,255);
                        cin.ignore();
                }
        }
//      mostrarPaleta();
//      cin.ignore();
}
```