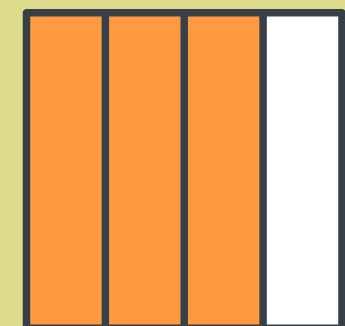
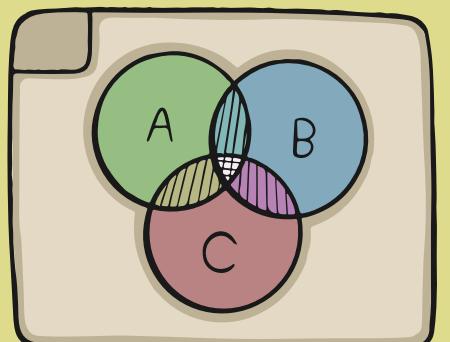
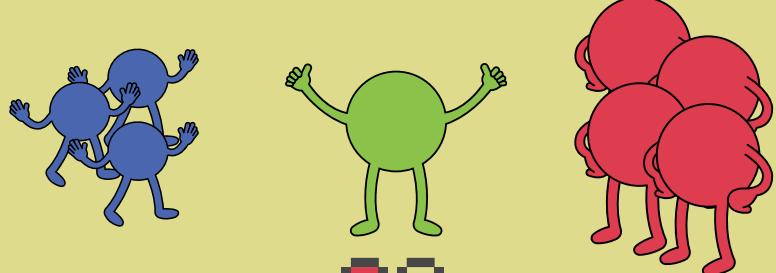
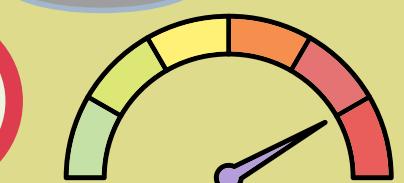




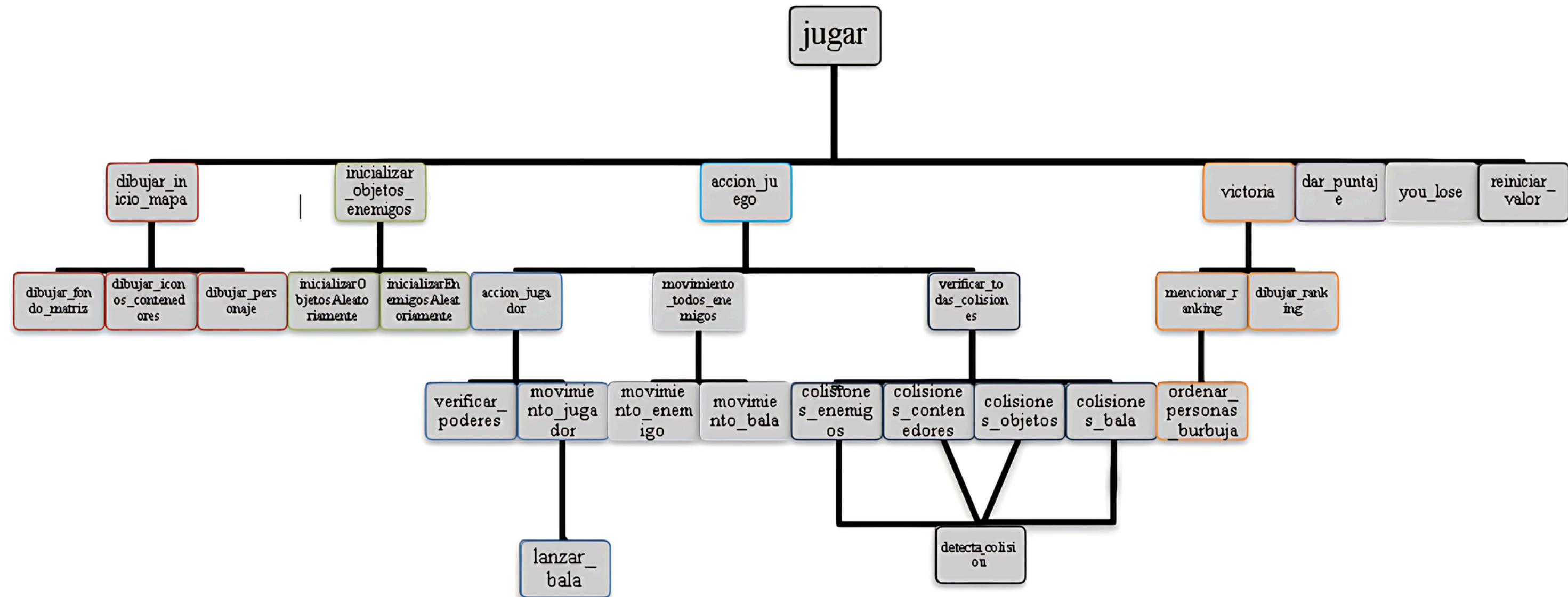
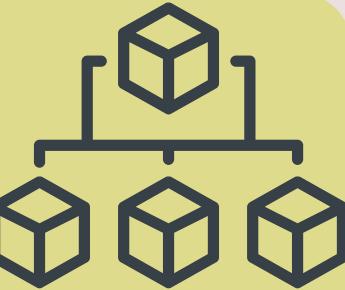
# Objetivo



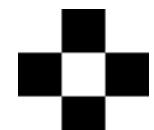
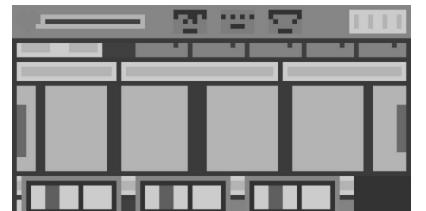
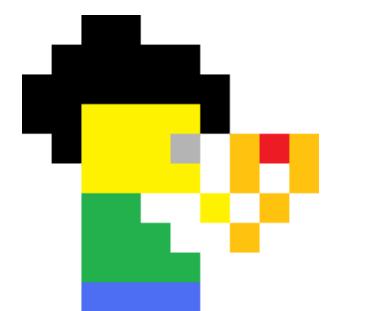
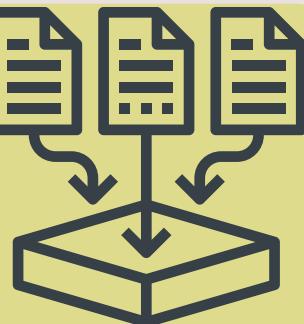
$$\frac{3}{4}$$



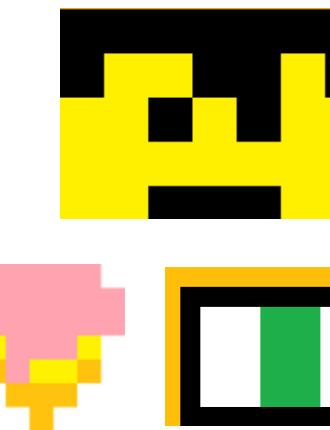
# Diagrama de módulos



# Estructura de datos



## Structs



```
typedef struct Lugar {
    Fondo fondo;
    Entidad personaje;
    string dibujo_bala;
    AtributosLugar atributos;
    Ranking ranking_lugar;

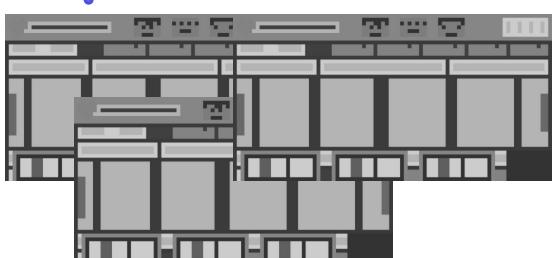
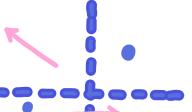
    Lugar(string** fondo, Entidad* personaje) : fondo(fondo), personaje(personaje) {}
```



```
typedef struct AtributosLugar {
    Entidad* enemigos;
    Entidad* enemigos_no_activos;
    Contenedor* contenedores;
    Objeto* objetos_suelo;
    Objeto* todos_objetos;
    Bala* balas;

    string texto_objeto_recogido, dibujo_objeto_recogido;
    string dibujo_bala_lugar = dibujo_bala;
    Tutor tutores[3] = { tutor_1, tutor_2, tutor_3 };
    int balas_cantidad = 0, balas_total = 1, enemigos_cantidad = 0;
    int cantidad_objetos_suelo, cantidad_objetos_total;
    int valor_objeto;
    bool llevando_objeto = false;
    int color_fondo_item, color_fondo_contenedor;
    AtributosLugar() : enemigos(), enemigos_no_activos(), contenedores(),
        todos_objetos(), balas(), texto_objeto_recogido(), dibujo_objeto_recogido(),
        tutores(), balas_cantidad(), balas_total(), enemigos_cantidad(),
        cantidad_objetos_suelo(), cantidad_objetos_total(), valor_objeto(),
        llevando_objeto(), color_fondo_item(), color_fondo_contenedor()
} AtributosLugar;
```

## Matrices



```
float posiciones_esquinas_enemigos[4][2] = { {5, 7}, {130, 7}, {5, 20}, {130, 20} };

int direcciones[4] = { 0, 1, 0, 1 };

for (int i = 0; i < numero_enemigos; i++) {
    // Selección aleatoria de enemigo
    int queEnemigo = rand() % 4; // 0 enemigo_1, 1 enemigo_2, 2 enemigo_3
    Entidad nuevo_enemigo = lugar.atributos.enemigos_no_activos[queEnemigo];
    nuevo_enemigo.posicion.x = posiciones_esquinas_enemigos[i][0];
    nuevo_enemigo.posicion.y = posiciones_esquinas_enemigos[i][1];
    nuevo_enemigo.estado.direccion = direcciones[i];
    nuevos_enemigos[i] = nuevo_enemigo;
    if (lugar.atributos.tutores[2].poder_activado) nuevos_enemigos[i].velocidad.rapidez = 0;
}
```

```
typedef struct Fondo {
    string** matriz;
    Fondo() : matriz() {}
    Fondo(string** matriz) : matriz(matriz) {}
} Fondo;
```

```
int posiciones[][3] = {
    {1, 7, 0}, {1, 8, 1},
    {1, 18, 2}, {1, 19, 3},
    {26, 21, 4}, {26, 22, 5},
    {26, 23, 6}, {26, 24, 7},
    {26, 25, 8}, {26, 26, 9},
    {26, 27, 10}, {26, 28, 11},
    {26, 30, 12}, {26, 31, 13},
    {26, 32, 14}, {26, 33, 15},
    {26, 34, 16}, {26, 35, 17},
    {26, 36, 18}, {26, 37, 19},
    {26, 38, 20}, {26, 39, 21},
    {26, 40, 22}, {26, 31, 23},
    {71, 32, 24}, {71, 33, 25},
    {71, 34, 26}, {71, 35, 27},
    {71, 36, 28}, {63, 37, 29},
    {63, 38, 30}, {63, 39, 31}
};
```



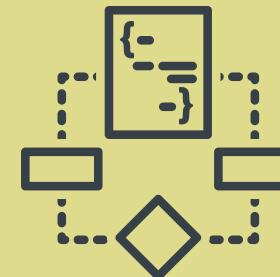
## Arreglos

```
Entidad* nuevos_enemigos = new Entidad[numero_enemigos];
```

```
contenedor_1[] = { "carnepodrida", "esenciafantasma", "cola", "slime", "ala", "hueso" };
```

```
Lugar bosque(matriz_fondos(fondos_bosque), agregar_enemigos_no_activos);
Lugar calabozo(matriz_fondos(fondos_mazmorra), agregar_enemigos_no_activos);
Lugar colegio(matriz_fondos(fondos_colegio), agregar_enemigos_no_activos);
Lugar lugares_juego[] = { bosque, calabozo, colegio };
```

# Algoritmos

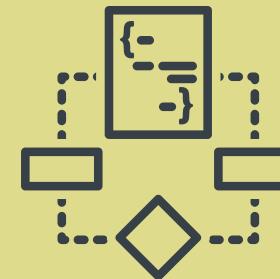


## movimiento\_jugador()

```
/*void movimiento_jugador(Lugar& lugar, bool& musica, bool& salir) {
    float pos_x_guardada = lugar.personaje.posicion.x, pos_y_guardada = lugar.personaje.posicion.y;
    int guardado_direccion = lugar.personaje.estado.direccion;
    if (_kbhit()) {
        char tecla = _getch();
        if (tecla == ARRIBA && lugar.personaje.posicion.y > 7) lugar.personaje.posicion.y -= lugar.personaje.velocidad.rapidez / FRAMES;
        else if (tecla == ABAJO && lugar.personaje.posicion.y + lugar.personaje.dimension.longitud_y < HEI)
        else if (tecla == IZQUIERDA && lugar.personaje.posicion.x >= 1) {
            lugar.personaje.posicion.x -= lugar.personaje.velocidad.rapidez / FRAMES;
            lugar.personaje.estado.direccion = 1;
        }
        else if (tecla == DERECHA && lugar.personaje.posicion.x + lugar.personaje.dimension.longitud_x <= lugar.personaje.dimension.longitud_x - lugar.personaje.velocidad.rapidez / FRAMES;
        lugar.personaje.estado.direccion = 0;
    }
    else if (tecla == ESPACIO && ((lugar.personaje.posicion.x + lugar.personaje.dimension.longitud_x - if (lugar.atributos.balas_cantidad < lugar.atributos.balas_total) lanzar_bala(lugar);
}
else if ((tecla == TECLA_1) && lugar.atributos.tutores[0].tutor_activado) {
    lugar.atributos.tutores[0].tutor_activado = false;
    lugar.atributos.balas_total = 3;
    lugar.atributos.tutores[0].poder_activado = true;
    for (int i = 0; i < 4; i++) lugar.atributos.tutores[0].cara_tutor[i] = 12;
}
else if ((tecla == TECLA_2) && lugar.atributos.tutores[1].tutor_activado) {
    lugar.atributos.tutores[1].tutor_activado = false;
}
```

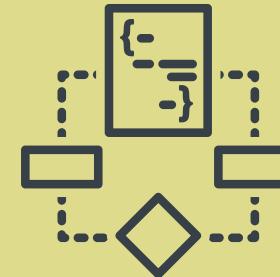
```
else if ((tecla == TECLA_2) && lugar.atributos.tutores[1].tutor_activado) {
    lugar.atributos.tutores[1].tutor_activado = false;
    lugar.personaje.estado.vida = 20;
    lugar.atributos.tutores[1].poder_activado = true;
    for (int i = 0; i < 4; i++) lugar.atributos.tutores[1].cara_tutor[i] = 10;
    dibujar_lineas(Rojo0, 14, 3, 40, i);
}
else if ((tecla == TECLA_3) && lugar.atributos.tutores[2].tutor_activado) {
    lugar.atributos.tutores[2].tutor_activado = false;
    lugar.atributos.tutores[2].poder_activado = true;
    for (int i = 0; i < 4; i++) lugar.atributos.tutores[2].cara_tutor[i] = 9;
    for (int i = 0; i < lugar.atributos.enemigos_cantidad; i++) lugar.atributos.enemigos[i].vivo = false;
}
else if (tecla == ESC) {
    salir = true;
}
else if (tecla == 77 || tecla == 109) {
    if (musica) {
        PlaySound(0, 0, 0);
        musica = false;
    }
    else {
        PlaySound(TEXT(".\\musica\\fondo.wav"), NULL, SND_FILENAME | SND_LOOP | SND_ASYNC);
        musica = true;
    }
}
: (int(pos_x_guardada) != int(lugar.personaje.posicion.x) || int(pos_y_guardada) != int(lugar.personaje.guardado_posicion.x = pos_x_guardada;
```

# Algoritmos



```
void lanzar_bala(Lugar& lugar) {
    lugar.atributos.balas_cantidad++;
    Bala* aumenta_balas = new Bala[lugar.atributos.balas_cantidad];
    for (int i = 0; i < lugar.atributos.balas_cantidad - 1; i++) aumenta_balas[i] = lugar.atributos.balas[i];
    Bala nueva_bala(lugar.personaje.posicion.x + lugar.personaje.dimension.longitud_x, lugar.personaje.posicion.y);
    if (lugar.personaje.estado.direccion == 1) nueva_bala.posicion.x = lugar.personaje.posicion.x - 3;
    aumenta_balas[lugar.atributos.balas_cantidad - 1] = nueva_bala;
    lugar.atributos.balas = aumenta_balas;
}
```

# Algoritmos



## inicializar\_objetos\_enemigos()

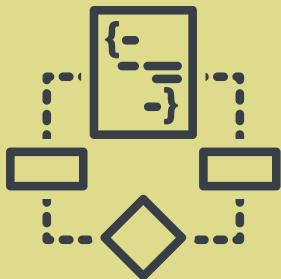
```
void inicializar_objetos_enemigos(Lugar &lugar) {
    if (lugar.atributos.cantidad_objetos_suelo == 0 && !lugar.atributos.llevando_objeto) inicializarObjetosAleatoriamente(lugar);
    if (lugar.atributos.enemigos_cantidad == 0 && !lugar.atributos.llevando_objeto) inicializarEnemigosAleatoriamente(lugar);
}
```

```
void inicializarObjetosAleatoriamente(Lugar& lugar) {
    int numero_objetos = 6;
    Objeto* nuevos_objetos = new Objeto[numero_objetos];
    int posiciones_esquinas_suministros_final[2][6][2] = {
        { {68, 18}, {78, 8}, {88, 28}, {48, 15}, {58, 28}, {58, 18} },
        { {58, 15}, {88, 16}, {34, 19}, {69, 28}, {73, 15}, {38, 14} }
    };
    int posiciones = rand() % 2;
    for (int i = 0; i < numero_objetos; i++) {
        // Selección aleatoria de objeto
        int queObjeto = rand() % lugar.atributos.cantidad_objetos_total; // 0 comida_1, 1 comida_2, 2 comida_3
        Objeto nuevo_objeto = lugar.atributos.todos_objetos[queObjeto];
        nuevo_objeto.posicion.x = posiciones_esquinas_suministros_final[posiciones][i][0];
        nuevo_objeto.posicion.y = posiciones_esquinas_suministros_final[posiciones][i][1];
        nuevos_objetos[i] = nuevo_objeto;
    }
    lugar.atributos.objetos_suelo = nuevos_objetos;
    lugar.atributos.cantidad_objetos_suelo = numero_objetos;
}
```

```
void inicializarEnemigosAleatoriamente(Lugar& lugar) {
    int numero_enemigos = 4;
    Entidad* nuevos_enemigos = new Entidad[numero_enemigos];
    float posiciones_esquinas_enemigos[4][2] = { {5, 7}, {138, 7}, {5, 28}, {138, 28} };
    int direcciones[4] = { 0, 1, 0, 1 };

    for (int i = 0; i < numero_enemigos; i++) {
        // Selección aleatoria de enemigo
        int queEnemigo = rand() % 4; // 0 enemigo_1, 1 enemigo_2, 2 enemigo_3
        Entidad nuevo_enemigo = lugar.atributos.enemigos_no_activos[queEnemigo];
        nuevo_enemigo.posicion.x = posiciones_esquinas_enemigos[i][0];
        nuevo_enemigo.posicion.y = posiciones_esquinas_enemigos[i][1];
        nuevo_enemigo.estado.direccion = direcciones[i];
        nuevos_enemigos[i] = nuevo_enemigo;
        if (lugar.atributos.tutores[2].poder_activado) nuevos_enemigos[i].velocidad.rapidez = 0;
    }
    lugar.atributos.enemigos = nuevos_enemigos;
    lugar.atributos.enemigos_cantidad = numero_enemigos;
}
```

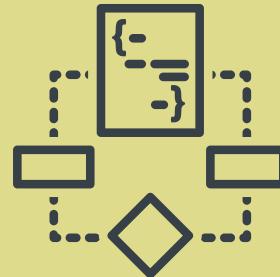
# Algoritmos



## colisiones\_contenedores()

```
void colisiones_contenedores(Lugar& lugar) {
    for (int i = 0; i < 3; i++) {
        if (detecta_colision(lugar.personaje.posicion.x, lugar.personaje.posicion.y, lugar.personaje.dimension.longitud_x, lugar.personaje.dimension.longitud_y)) {
            bool verificar_objetos = false;
            lugar.atributos.contenedores[i].atributos.color_letra = RojoC;
            for (int j = 0; j < lugar.atributos.contenedores[i].atributos.items_total; j++) {
                if (lugar.atributos.contenedores[i].atributos.objetos[j] == lugar.atributos.texto_objeto_recogido) {
                    verificar_objetos = true;
                    lugar.atributos.contenedores[i].atributos.color_letra = VerdeC;
                    break;
                }
            }
            if (verificar_objetos) lugar.atributos.contenedores[i].atributos.items_ahora += lugar.atributos.valor_objeto;
            else lugar.atributos.contenedores[i].atributos.items_ahora -= lugar.atributos.valor_objeto;
            if (lugar.atributos.contenedores[i].atributos.items_ahora < 0) lugar.atributos.contenedores[i].atributos.items_ahora = 0;
            else if (lugar.atributos.contenedores[i].atributos.items_ahora > lugar.atributos.contenedores[i].atributos.fracciones[0].todo * lugar.atributos.contenedores[i].atributos.fracciones[0].fraccion) lugar.atributos.contenedores[i].atributos.items_ahora = lugar.atributos.contenedores[i].atributos.fracciones[0].todo * lugar.atributos.contenedores[i].atributos.fracciones[0].fraccion;
            dibujar_lineas(lugar.atributos.color_fondo_item, 146, 37, 28, 7);
            dibujar_lineas(lugar.atributos.color_fondo_contenedor, lugar.atributos.contenedores[i].posicion.x + 3, lugar.atributos.contenedores[i].posicion.y + 3, 1);
            lugar.atributos.texto_objeto_recogido = "";
            lugar.atributos.llevando_objeto = false;
            lugar.atributos.valor_objeto = 0;
            int x_temporal = WIDTH - 22, y_temporal = 9 + i * 9, total_x = 0, medio_x = WIDTH - 11;
            for (int x = 0; x < lugar.atributos.contenedores[i].atributos.cantidad_fracciones; x++) {
                total_x += lugar.atributos.contenedores[i].atributos.fracciones[x].largo + 1;
                if (x == 0) total_x -= 1;
            }
        }
    }
}
```

# Algoritmos



```
medio_x -= total_x;

for (int j = 0; j < lugar.atributos.contenedores[i].atributos.cantidad_fracciones; j++) {
    string modificado = lugar.atributos.contenedores[i].atributos.fracciones[j].dibujo;
    for (int z = 0, letra = 0; z < ((lugar.atributos.contenedores[i].atributos.items_ahora - (j * lugar.atributos.contenedores[i].atributos.fracciones[j].todo)))
        bool cambio = false;
        if (modificado[letra] == '/') {
            letra++;
            char caracter;
            caracter = modificado[z - lugar.atributos.contenedores[i].atributos.fracciones[j].largo] == 'A' ? 'B' : 'A';
            modificado[letra] = caracter;
        }
        else if (letra - 1 < 0) modificado[letra] = 'A';
        else {
            if (modificado[letra - 1] == 'A') modificado[letra] = 'B';
            else modificado[letra] = 'A';
        }
        letra++;
    };
    dibujar_personaje(duplicar_string(modificado), medio_x, y_temporal + 2);
    medio_x += lugar.atributos.contenedores[i].atributos.fracciones[j].largo * 2 + 2;
}
break;
```

# DEMO



[https://youtu.be/o-0iuWVV\\_Ow?  
si=JROK\\_PxYcvKYPi5X](https://youtu.be/o-0iuWVV_Ow?si=JROK_PxYcvKYPi5X)

# Conclusiones

- El juego integra a la matemática como una de sus mecánicas principales.
- El programa cuenta con 7 ramas principales, las cuales han sido divididas para poder resolverlas. Se aprovecha la programación modular.
- Se han utilizado de la mejor manera las diferentes tipos de estructuras vistas hasta el momento, para lograr armar la base de cada nivel y juego.



**Gracias!**