

# ~~EDWORLD: SISTEMA DE GESTIÓN DE MUNDOS 3D~~

Modelo de Desarrollo de Programas  
Práctica Curso 2008-2009

**Grupo 38:**  
Raúl Gómez Pascual – 040292  
José María Rico Pérez – 050129  
José Antonio Rodríguez Martínez – 050133  
Gema Molina Vaquero – 040128

**PARTE A: Modelo Orientado a Objetos**

1.	Captura de Requisitos:	
1.1.	Diagrama de casos de uso.....	3
1.2.	Breve descripción de todos los actores y casos de uso.....	4
1.3.	Diagrama de transición entre estados y descripción completa de tres casos de uso.....	5
1.3.1.	Agregar objeto	
1.3.2.	Agregar triángulo a objeto	
1.3.3.	Agregar acción de transformación	
2.	Análisis y diseño:	
2.1.	Diagrama final de clases.....	9
2.2.	Diagrama de secuencia de cada caso de uso en la versión final del diseño.....	10
2.3.	Representación completa de cada clase.....	19
3.	Implementación:	
3.1.	Diagrama de componentes.....	24
3.2.	Codificación en el lenguaje elegido.....	24

**PARTE B: Modelo Orientado al Flujo de Datos**

1.	Ánálisis:	
1.1.	Modelo ambiental.....	26
1.2.	Modelo de comportamiento:	
1.2.1.	Modelo de proceso .....	27
1.2.2.	Modelo de datos: diagrama E/R.....	32
1.2.3.	Diagrama de transición entre estados .....	33
2.	Diseño:	
2.1.	Modelo de implantación de programas:	
2.1.1.	Diagrama de estructura refinado.....	34
2.1.2.	Diagrama estructurado de todos los módulos .....	35

**modelo**

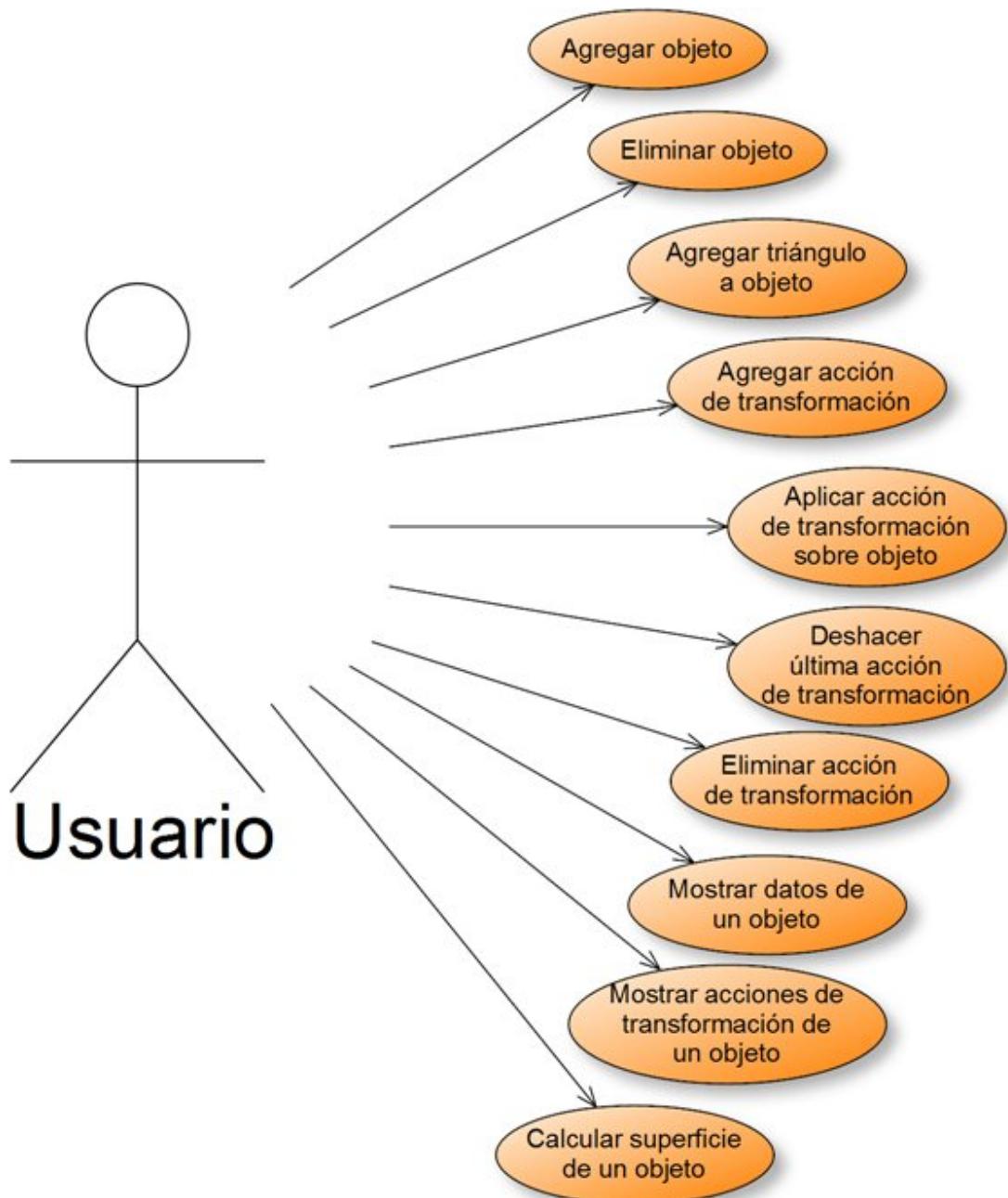
**ORIENTADO A**

**OBJETOS**

## 1. CAPTURA DE REQUISITOS

### 1.1. Diagrama de casos de uso:

En el diagrama podemos observar como el usuario puede efectuar cada uno de los distintos casos de uso:



## 1.2. Breve descripción de todos los actores y casos de uso:

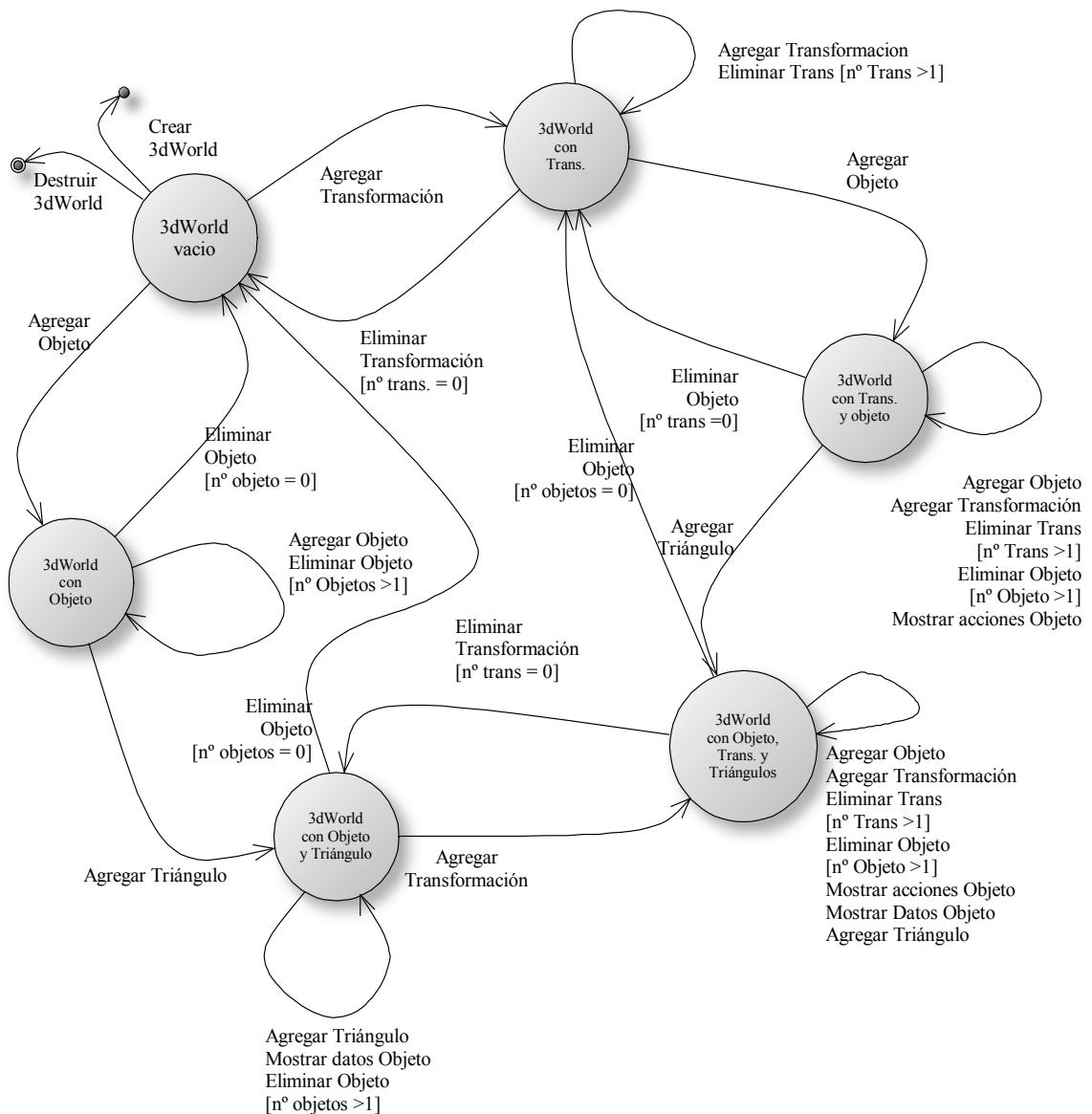
Durante la ejecución del programa sólo tenemos un actor, que llamamos “Usuario”. Este usuario introduce los comandos por el fichero de entrada y los lee en el fichero de salida, tras su ejecución.

A continuación describimos los distintos casos de uso:

1. **Agregar objeto:** El usuario solicita agregar un objeto. Una vez introducidos todos los datos procederemos a introducir un nuevo objeto siempre y cuando no exista otro objeto con el mismo nombre.
2. **Eliminar objeto:** El usuario solicita eliminar un objeto. Con el nombre del objeto, al eliminarlo lo que hacemos es eliminar todos los triángulos que lo componen.
3. **Agregar triángulo a objeto:** El usuario solicita agregar un triángulo a un objeto. Tomando sus datos y el nombre del objeto agregamos un nuevo triángulo mientras que éste no exista ya en el objeto, es decir, en un objeto no puede haber dos triángulos iguales. Además, para poder agregar el triángulo éste debe de tener sus tres puntos distintos.
4. **Agregar acción de transformación:** El usuario solicita agregar una acción de transformación a un objeto. Teniendo todos sus datos, agregamos una nueva acción si el identificador de ésta no existe todavía, incluso cuando el tipo de la transformación sea distinto, si el identificador es el mismo, no la agregamos.
5. **Aplicar acción de transformación sobre objeto:** El usuario solicita aplicar una acción de transformación sobre un objeto. Usamos el identificador de la transformación y el nombre del objeto donde quiero aplicar dicha transformación a todos los triángulos. Es importante llevar un orden de las transformaciones que vamos aplicando ya que posteriormente tendremos que deshacer la última.
6. **Deshacer la última acción de transformación:** El usuario solicita deshacer la última acción de transformación sobre un objeto. Conocido el nombre del objeto sobre el que queremos deshacer la última transformación, la deshacemos, borrando posteriormente esta acción de las aplicadas sobre ese objeto.
7. **Eliminar acción de transformación:** El usuario solicita eliminar una acción de transformación. A partir del identificador de ésta, se eliminará cualquier acción que haya sido previamente deshecha, ninguna otra.
8. **Mostrar datos de objeto:** El usuario solicita mostrar los datos de un objeto. Dado el nombre del objeto se mostrará el nombre de este, su color, el número de acciones de transformación aplicadas, el número de triángulos y los datos de cada uno de estos según su orden de creación.
9. **Mostrar acciones de transformación de un objeto:** El usuario solicita mostrar las acciones de transformación de un objeto. Dado el nombre del objeto se mostrará el nombre de este y cada una de sus acciones de transformación según su orden de aplicación. Se mostrará identificador y datos por cada transformación.
10. **Calcular superficie de un objeto:** El usuario solicita calcular la superficie de un objeto. Tomando el nombre del objeto se mostrará el nombre de este y su superficie (que es la suma de las superficies de todos sus triángulos).

**1.3. Diagrama de transición entre estados y descripción completa de al menos tres casos de uso:**

**Diagrama de transición de estados:**



### 1.3.1. Agregar objeto.

#### Pre condición:

Para poder agregar un objeto, previamente el usuario debe haber introducido el comando AOB, por el fichero de entrada. Una vez el programa haya reconocido este comando, procederá a leer los dos siguientes parámetros: nombre <no> y color <co>. Entonces ejecutamos agregar objeto que recibirá estos dos parámetros.

#### Flujo de eventos del camino básico:

1. El sistema está esperando la orden de usuario
2. Se piden los datos necesarios para la operación, en este caso, nombre <no> y color <co>.
3. A continuación se comprueba que no exista ningún objeto con ese nombre en el mundo 3d y que los datos introducidos sean correctos. En tal caso se agrega dicho objeto, ya que la operación se ha realizado satisfactoriamente.

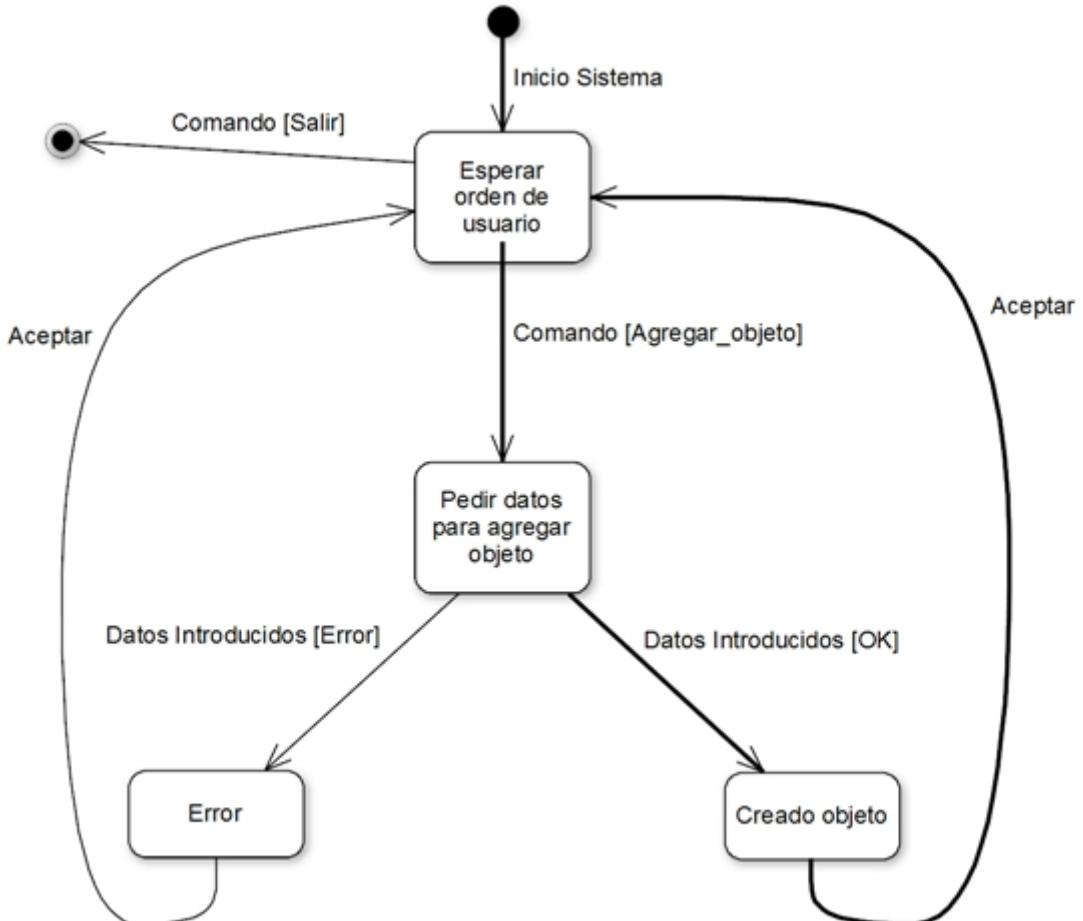
En caso de error se devolvería el mensaje de error correspondiente.

#### Pos condición:

El mundo 3d tiene un objeto más o el mismo número de objetos en caso de que objeto ya existiera.

#### Atributos:

- Nombre <no>
- Color <co>



### 1.3.2. Eliminar objeto.

#### Pre condición:

Para poder eliminar un objeto, previamente el usuario debe haber introducido el comando EOB, por el fichero de entrada. Una vez el programa haya reconocido este comando, procederá a leer el siguiente parámetro: el nombre del objeto que quiero eliminar <no>. Entonces ejecutamos eliminar objeto que recibirá este parámetro.

#### Flujo de eventos del camino básico:

1. El sistema está esperando la orden de usuario.
2. Se piden los datos necesarios para la operación, en este caso, nombre <no>.
3. A continuación se comprueba que exista algún objeto con ese nombre en el mundo 3d y que este dato introducido sea correcto. En tal caso se elimina dicho objeto, ya que la operación se ha realizado satisfactoriamente.

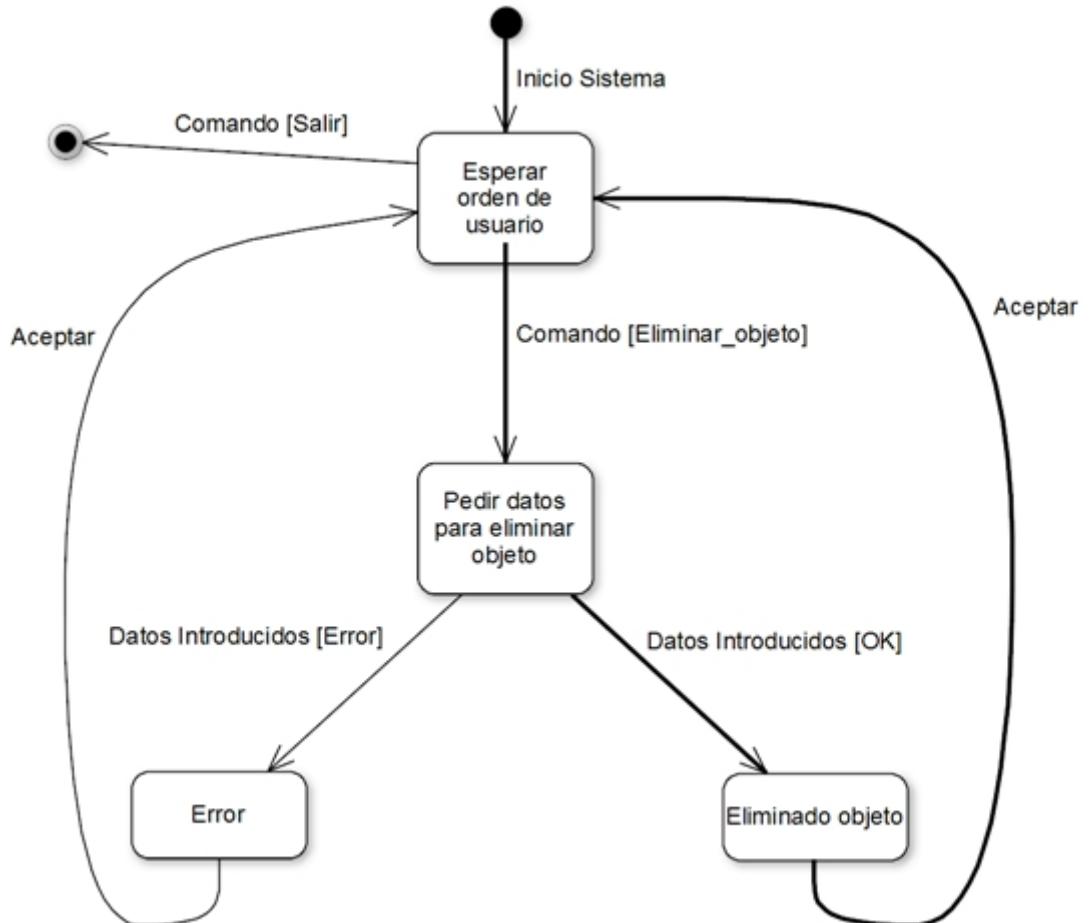
En caso de error se devolvería el mensaje de error correspondiente.

#### Pos condición:

El mundo 3d tiene un objeto menos o el mismo número de objetos en caso de que el objeto no existiera.

#### Atributos:

- Nombre <no>



### 1.3.3. Aplicar acción.

#### Pre condición:

Para poder aplicar una acción de transformación, previamente el usuario debe haber introducido el comando APA, por el fichero de entrada. Una vez el programa haya reconocido este comando, procederá a leer los siguientes parámetros: el nombre del objeto sobre el que quiere aplicar la acción *<no>* y el identificador de esa acción *<ia>*. Entonces ejecutamos aplicar acción que recibirá estos dos parámetros.

#### Flujo de eventos del camino básico:

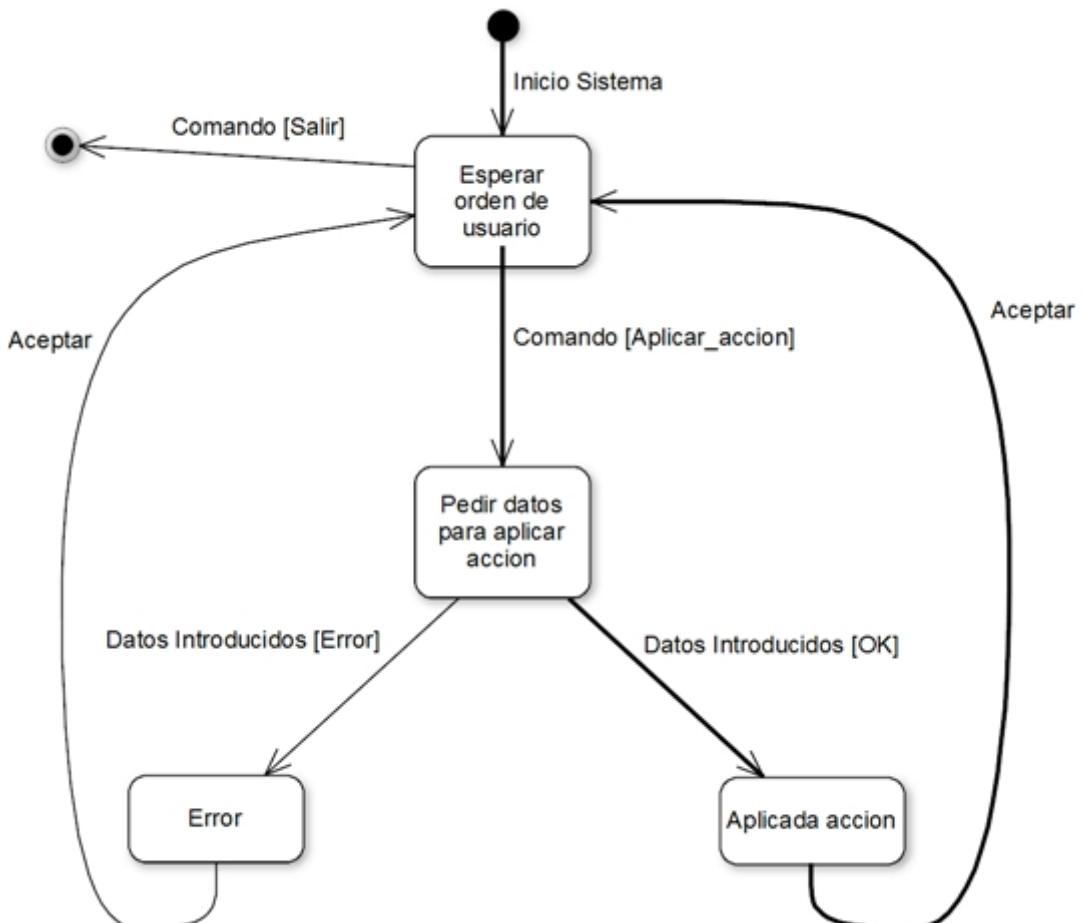
1. El sistema está esperando la orden de usuario.
2. Se piden los datos necesarios para la operación, en este caso, nombre *<no>* e identificador de acción *<ia>*.
3. A continuación se comprueba que exista algún objeto con ese nombre en el mundo 3d y que exista ese identificador de acción. En tal caso se aplica la acción de transformación a dicho objeto, ya que la operación se ha realizado satisfactoriamente. En caso de error se devolvería el mensaje de error correspondiente.

#### Pos condición:

Un objeto del mundo 3d tiene aplicada una transformación en su lista de triángulos, siempre que la lista de triángulos no fuera vacía.

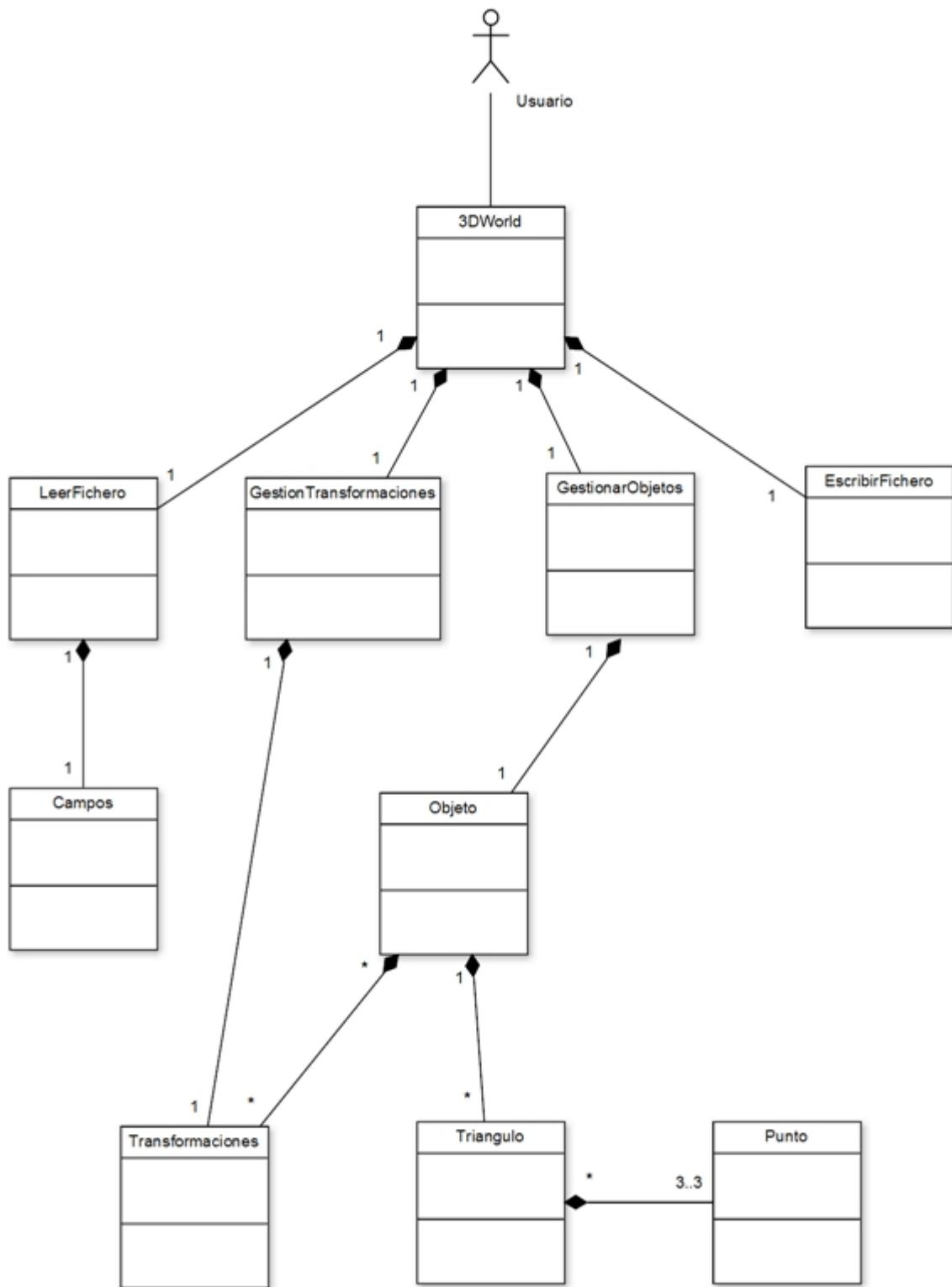
#### Atributos:

- Nombre *<no>*
- Identificador de acción *<ia>*



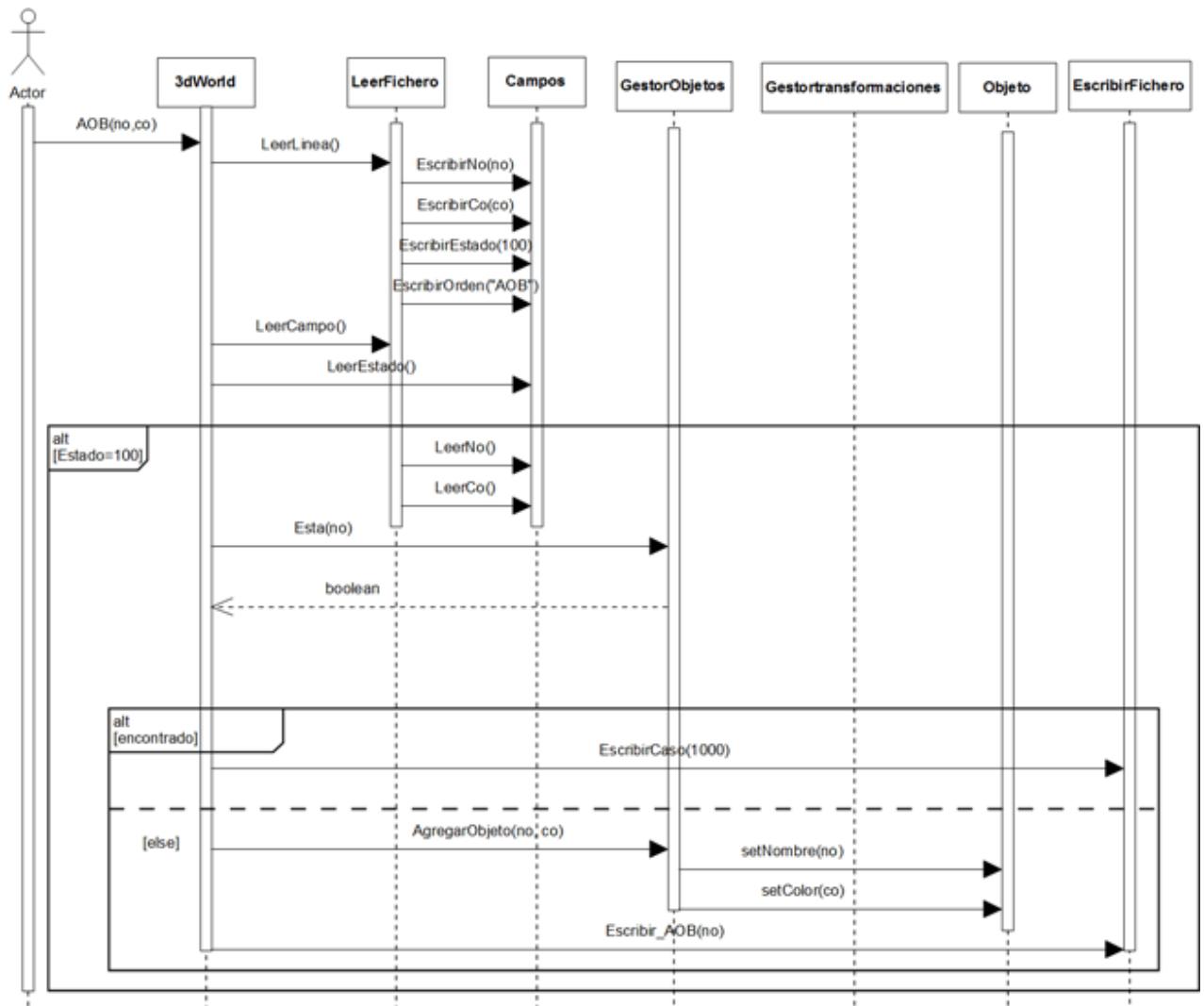
## 2. ANÁLISIS Y DISEÑO

### 2.1. Diagrama final de clases:

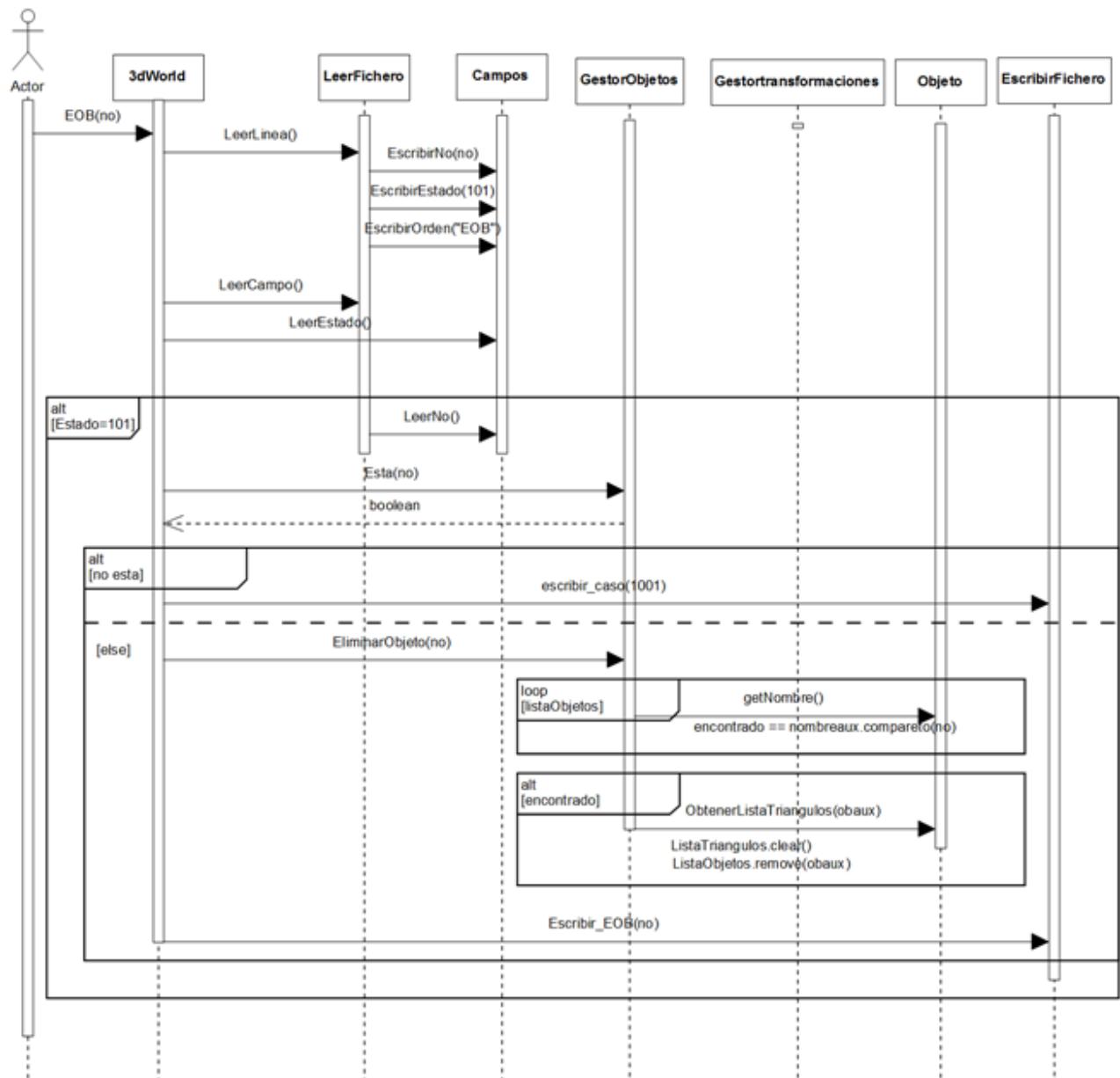


2.2. Diagrama de secuencia de cada caso de uso en la versión final del diseño:

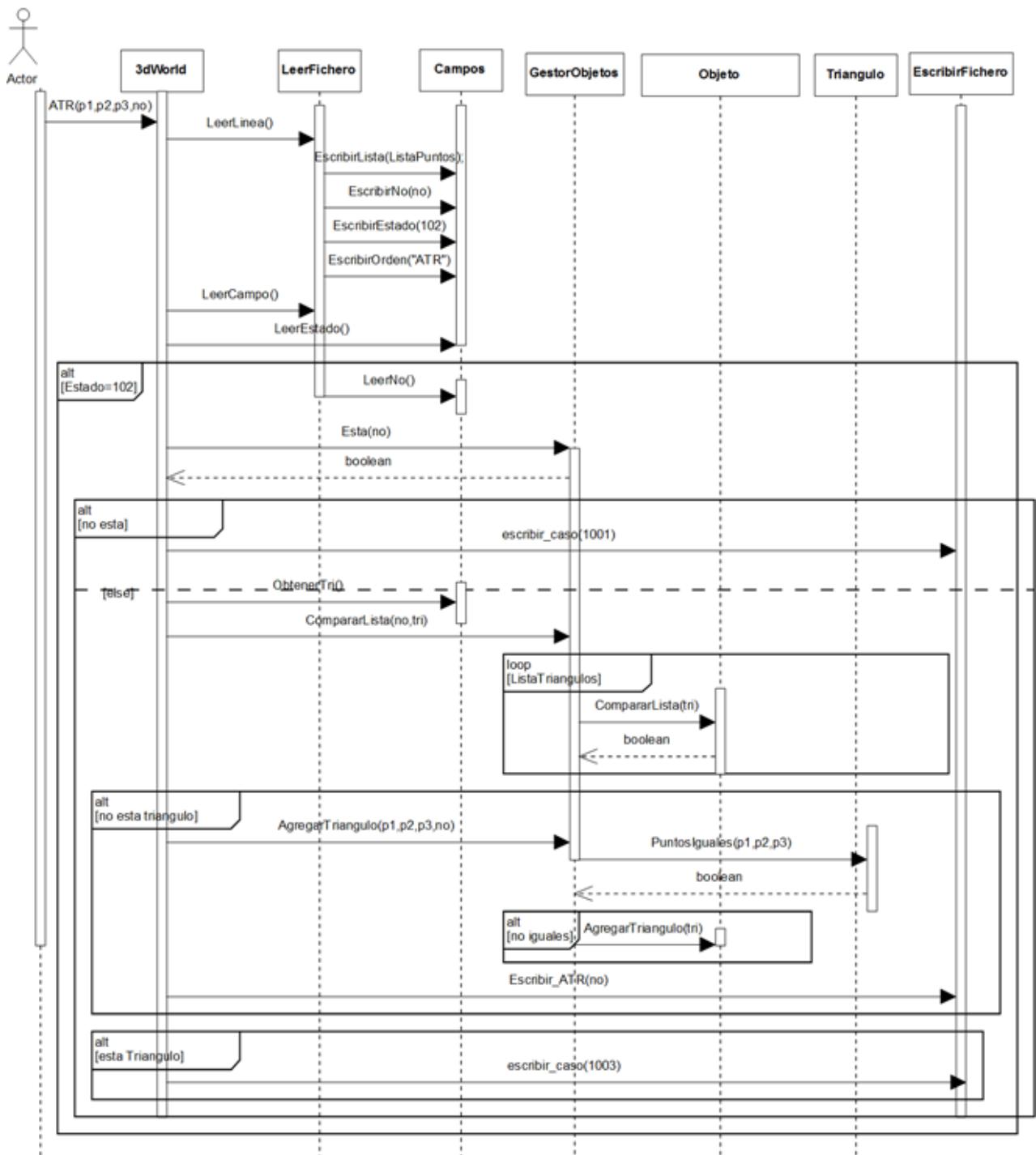
- Agregar objeto (<no>, <co>):



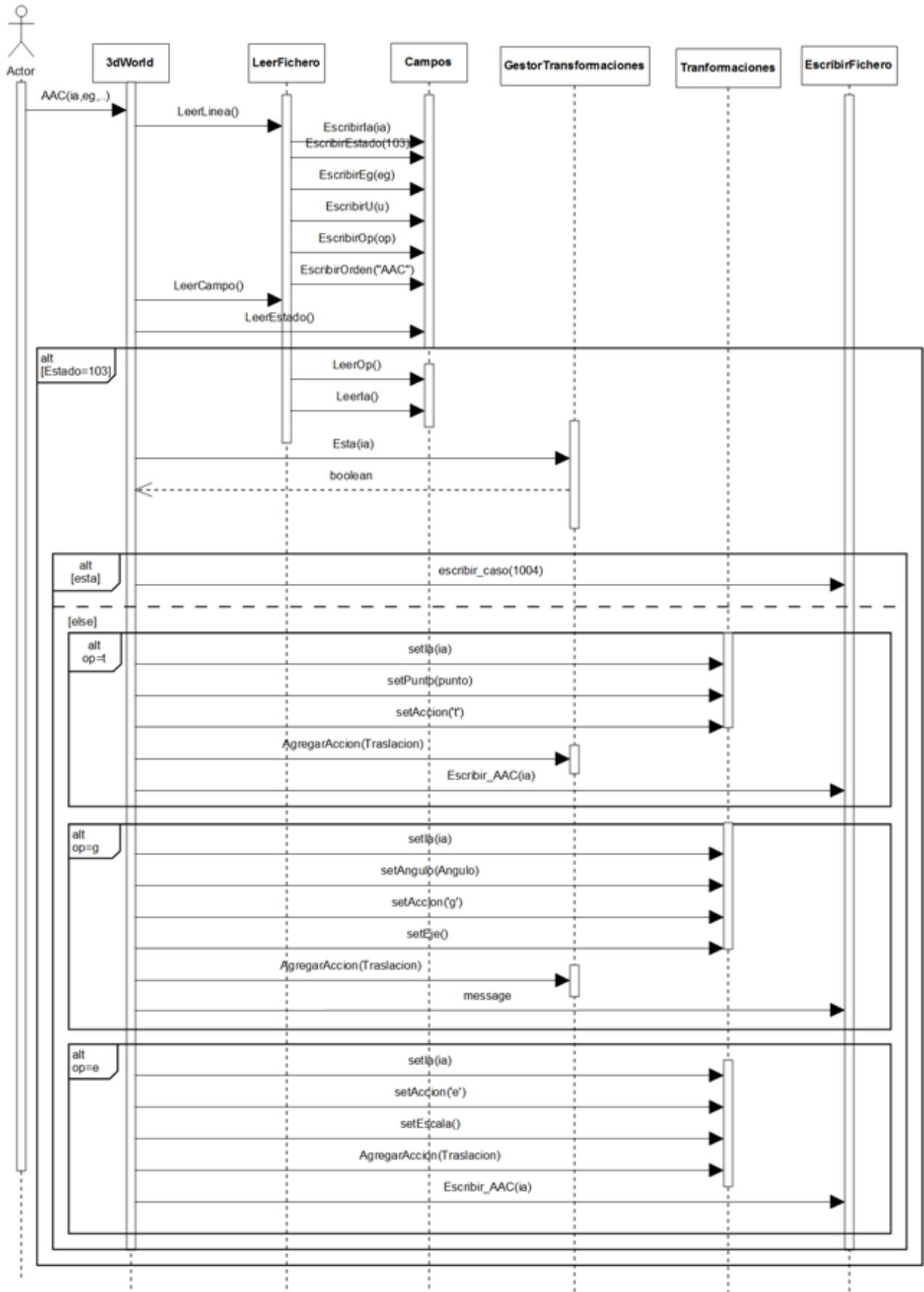
- Eliminar objeto (<no>):



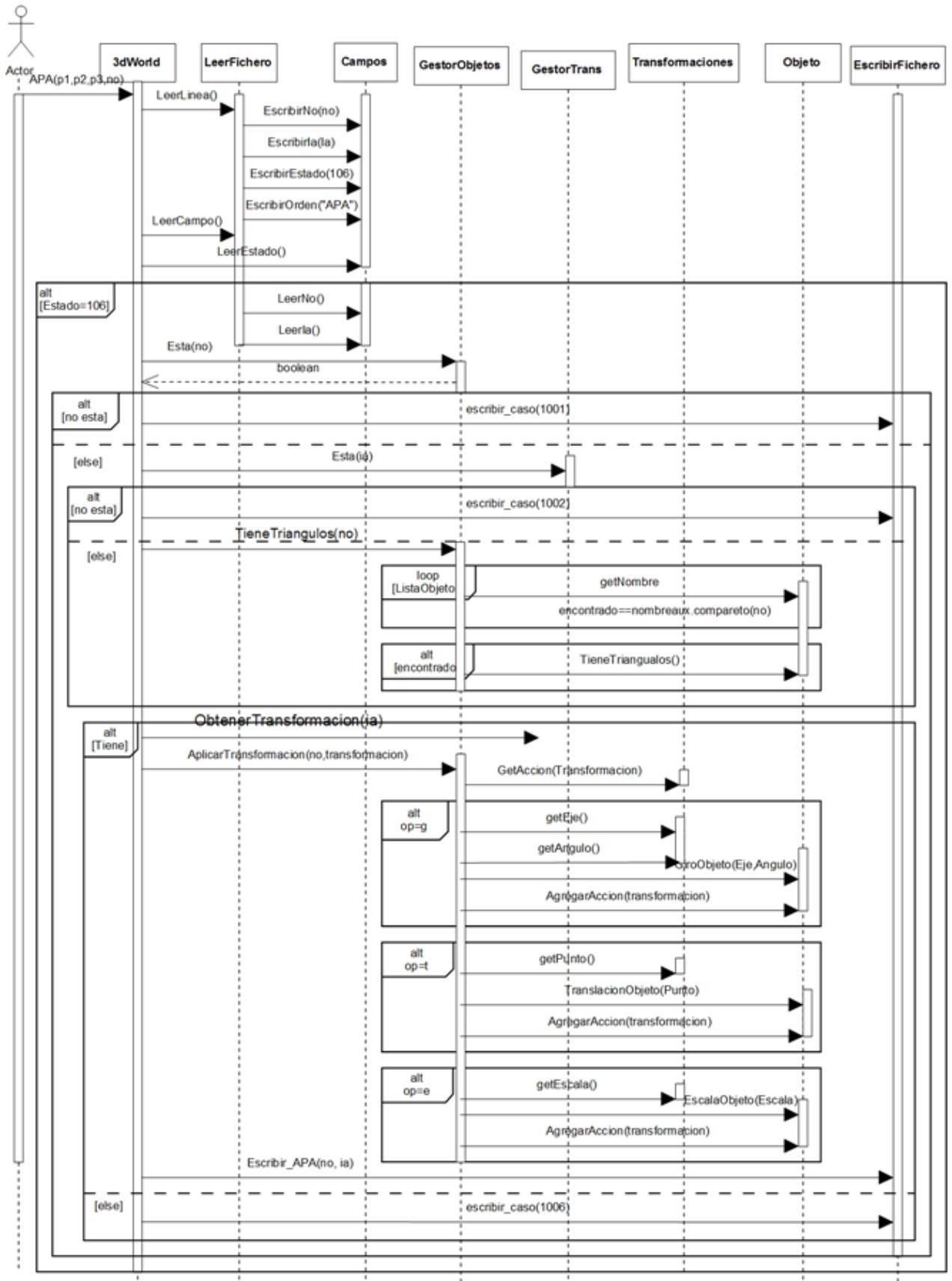
- Agregar triángulo a objeto (<x1>, <y1>, <z1>, <x2>, <y2>, <z2>, <x3>, <y3>, <z3>, <no>):



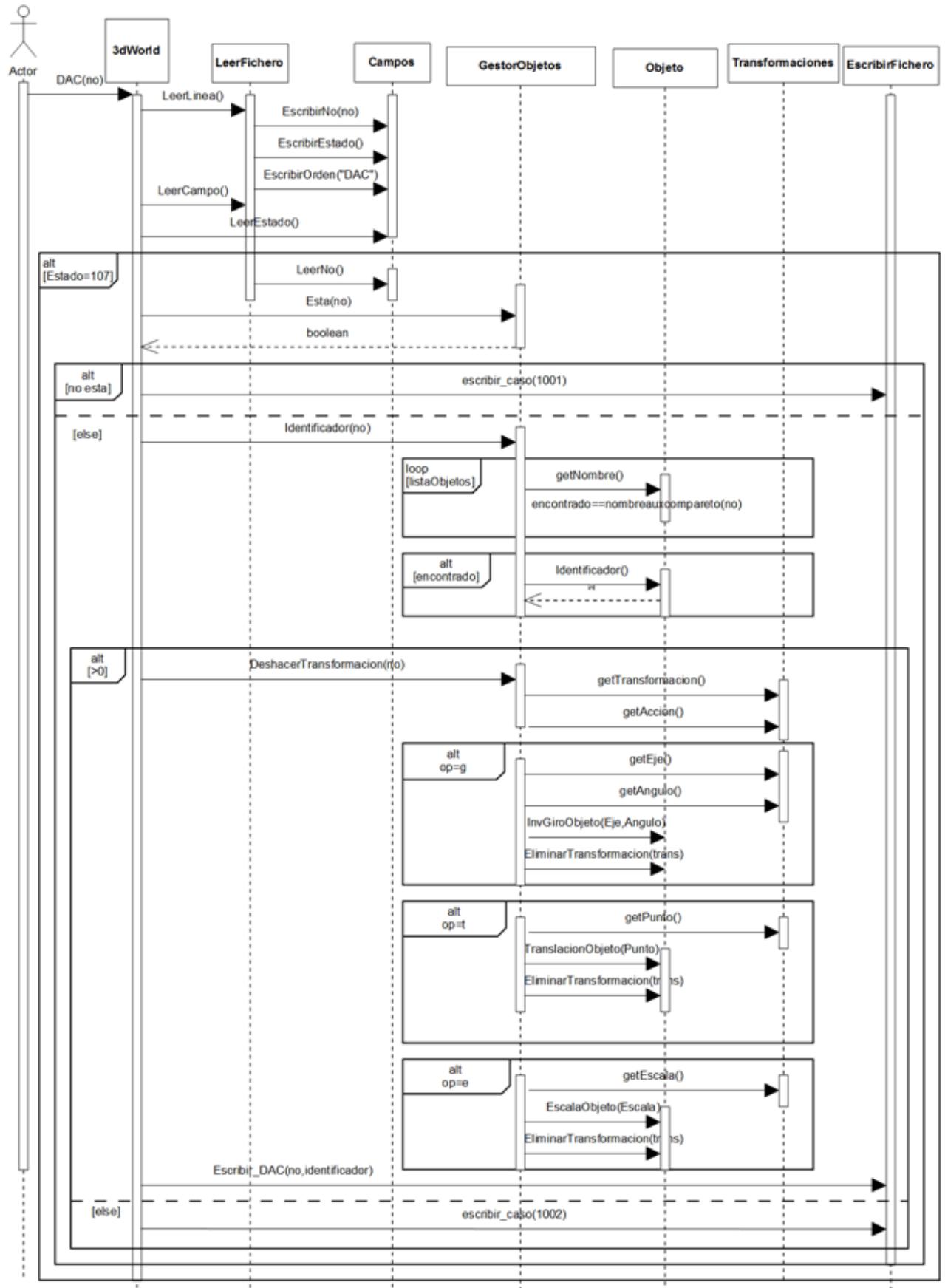
- Agregar acción de transformación (<ia>):



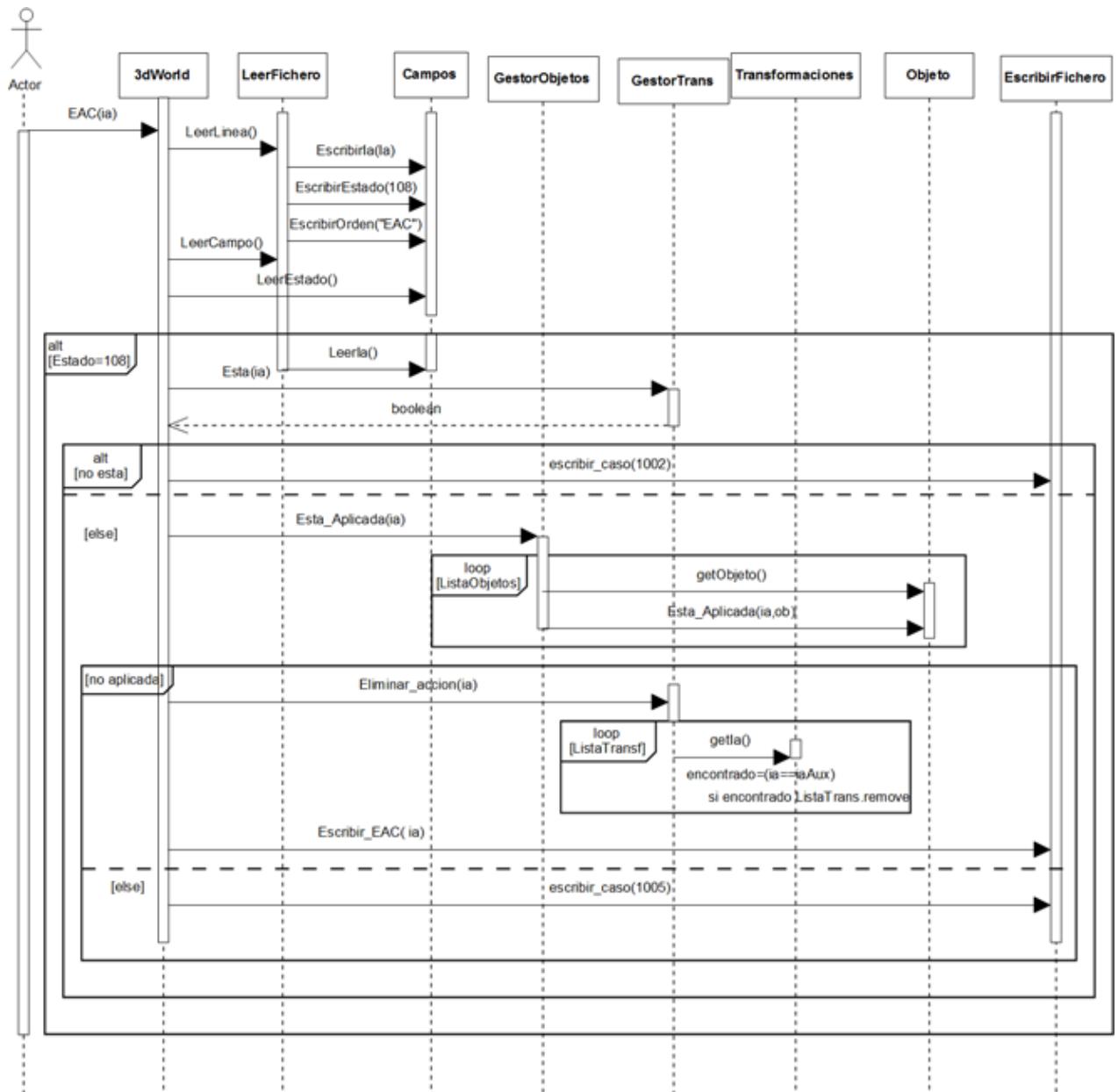
- Aplicar acción de transformación sobre objeto (<no>, <ia>):



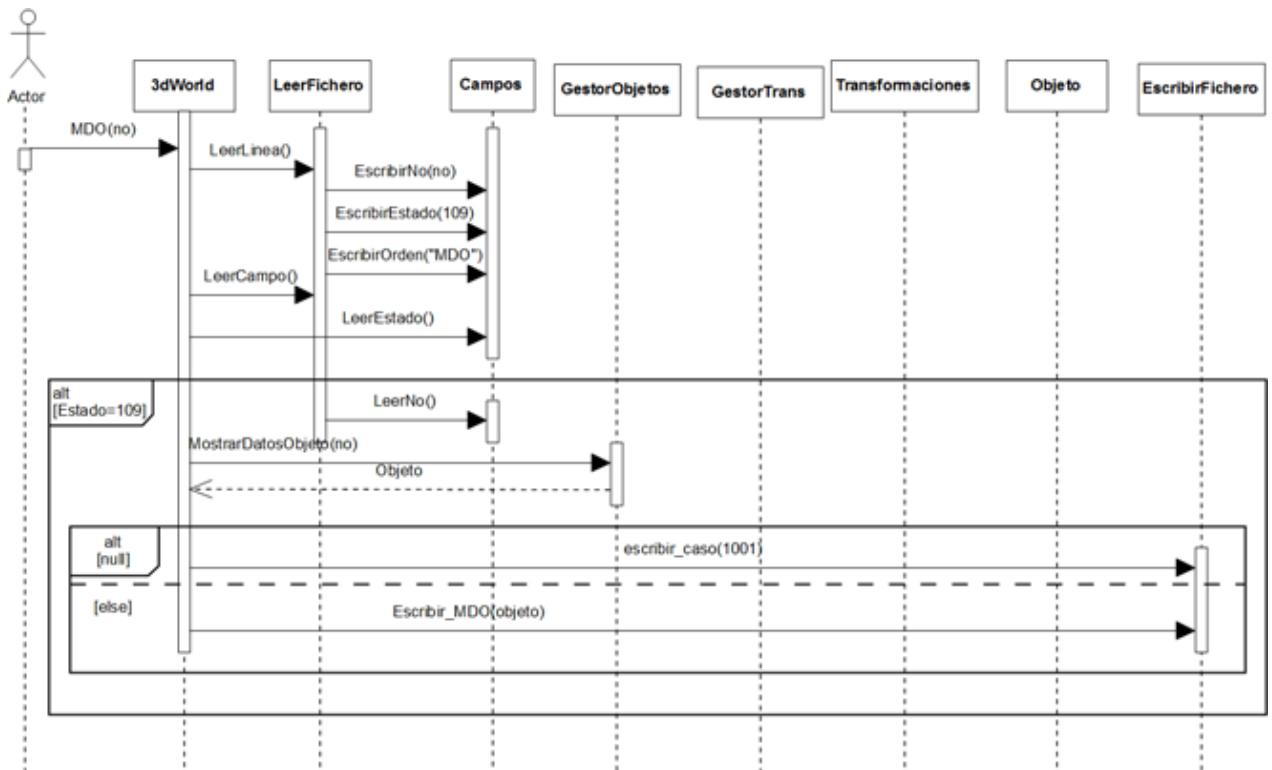
- Deshacer la última acción de transformación (<no>):



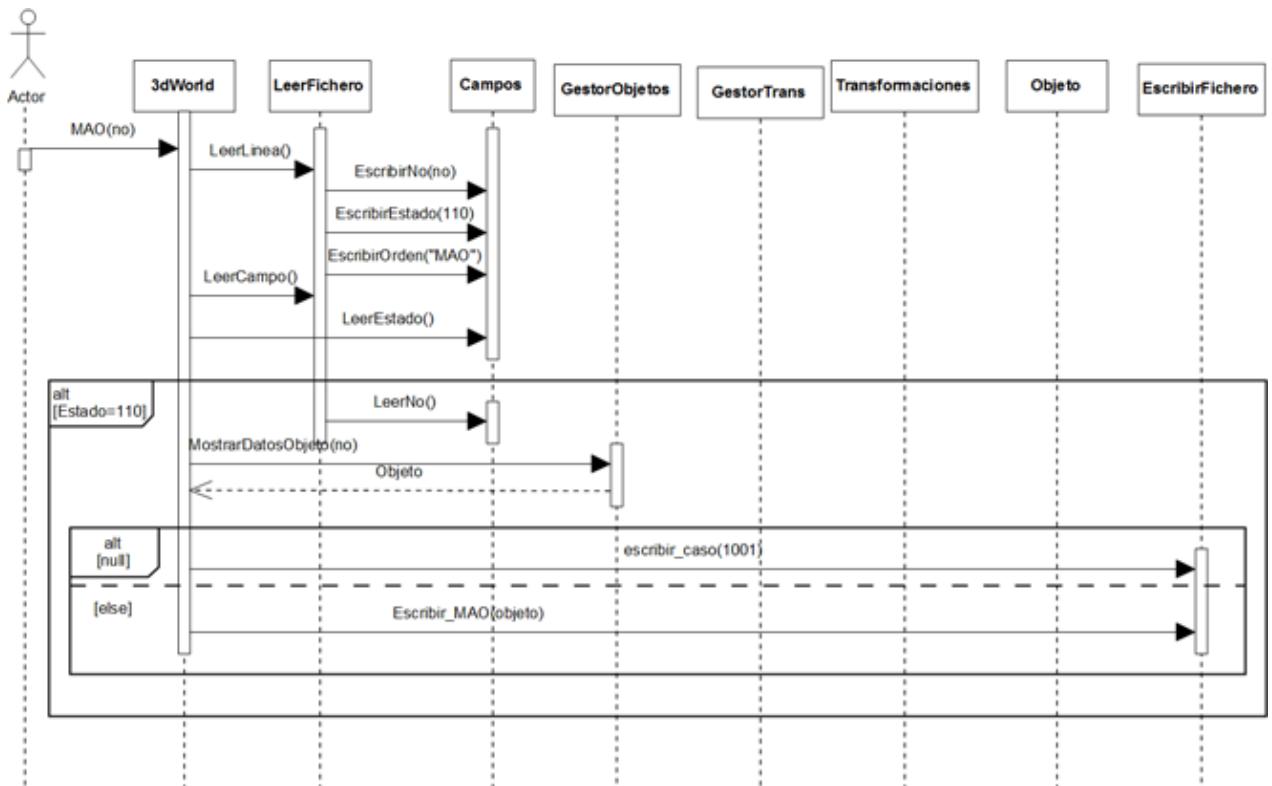
- Eliminar acción de transformación (<ia>):



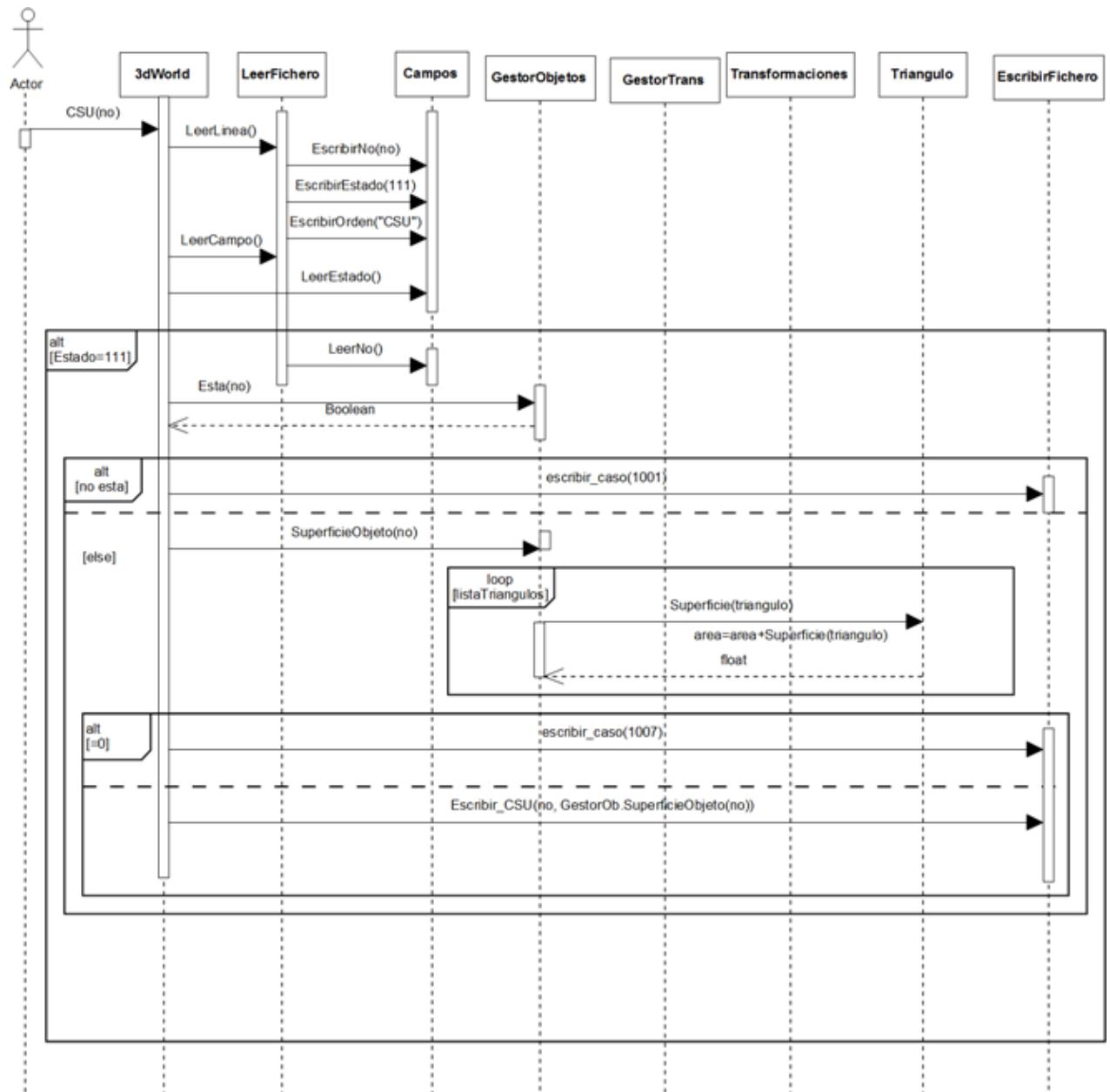
- Mostrar datos de objeto (<no>):



- Mostrar acciones de transformación de un objeto (<no>):



- Calcular superficie de un objeto (<no>):



### 2.3. Representación completa de cada clase

- Punto:

Punto
- x : float - y : float - z : float
+ getX () : float + setX (x : float) + getY () : float + setY (x : float) + getZ () : float + setZ (x : float) + PuntosIguales (p2 : Punto) : boolean + TraslacionPunto (punto : Punto) + InvTrasPunto (punto : Punto) + EscalaPunto (escala : float) + InvEscalaPunto (escala : float) + GiroPunto(eje : char, angulo : int) + InvGiroPunto(eje : char, angulo : int)

- Triángulo:

Triangulo
- punto1 : Punto - punto2 : Punto - punto3 : Punto
+ Triangulo () + getPunto1 () : Punto + setPunto1 (punto1 : Punto) + getPunto2 () : Punto + setPunto2 (punto2 : Punto) + getPunto3 () : Punto + setPunto3 (punto3 : Punto) + TriangulosIguales (tri2 : Triangulo) : boolean + PuntosIgualesTri (tri : Triangulo) : boolean + TraslacionTriangulo (punto : Punto) + EscalaTriangulo (escala : float) + GiroTriangulo (eje : char, angulo : int) + InvTraslacionTriangulo (punto : Punto) + InvEscalaTriangulo (escala : float) + InvGiroTriangulo (eje : char, angulo : int)

- Transformaciones:

Transformaciones
<ul style="list-style-type: none"> <li>- ia : int</li> <li>- accion : char</li> <li>- eje : char</li> <li>- angulo : int</li> <li>- punto : Punto</li> <li>- escala : float</li> </ul> <ul style="list-style-type: none"> <li>+ getIa () : int</li> <li>+ setIa (ia : int)</li> <li>+ getAccion () : char</li> <li>+ setAccion (accion : char)</li> <li>+ getEje () : char</li> <li>+ setEje (eje : char)</li> <li>+ getAngulo () : int</li> <li>+ setAngulo (angulo : int)</li> <li>+ getValorEscala () : float</li> <li>+ setEscala (escala : float)</li> <li>+ getPunto () : Punto</li> <li>+ setPunto (punto : Punto)</li> </ul>

- Objeto:

Objeto
<ul style="list-style-type: none"> <li>- Nombre : String</li> <li>- Color : String</li> <li>- ListaTriangulos : LinkedList &lt;Triangulo&gt;</li> <li>- ListaTransformaciones : LinkedList &lt;Transformaciones&gt;</li> </ul> <ul style="list-style-type: none"> <li>+ Objeto ()</li> <li>+ getNombre () : String</li> <li>+ setNombre (Nombre : String)</li> <li>+ getColor () : String</li> <li>+ setColor (Color : String)</li> <li>+ getListaTriangulos () : LinkedList &lt;Triangulo&gt;</li> <li>+ setListaTriangulos (ListaTriangulos : LinkedList &lt;Triangulo&gt;)</li> <li>+ getListaTransformaciones () : LinkedList &lt;Transformaciones&gt;</li> <li>+ setListaTransformaciones (ListaTransformaciones : LinkedList &lt;Transformaciones&gt;)</li> <li>+ CompararLista (Tri : Triangulo) : Boolean</li> <li>+ TraslacionObjeto (punto : Punto)</li> <li>+ InvTraslacionObjeto (punto : Punto)</li> <li>+ EscalaObjeto (escala : float)</li> <li>+ InvEscalaObjeto (escala : float)</li> <li>+ GiroObjeto (eje : char, angulo : int)</li> <li>+ InvGiroObjeto (eje : char, angulo : int)</li> <li>+ TieneTriangulos () : int</li> </ul>

- Campos:

Campos
<ul style="list-style-type: none"><li>- ListaPuntos : LinkedList &lt;String&gt;</li><li>- co : String</li><li>- dx : String</li><li>- dy : String</li><li>- dz : String</li><li>- eg : String</li><li>- ia : String</li><li>- no : String</li><li>- u : String</li><li>- ve : String</li><li>- x : String</li><li>- y : String</li><li>- z : String</li><li>- orden : String</li><li>- estado : int</li><li>- operacion : String</li></ul>
<ul style="list-style-type: none"><li>+ Campos ()</li><li>+ LeerCo () : String</li><li>+ EscribirCo (co : String)</li><li>+ LeerDx () : String</li><li>+ EscribirDx (dx : String)</li><li>+ LeerDy () : String</li><li>+ EscribirDy (dy : String)</li><li>+ LeerDz () : String</li><li>+ EscribirDz (dz : String)</li><li>+ LeerEg () : String</li><li>+ EscribirEg (eg : String)</li><li>+ Leerla () : String</li><li>+ Escribirla (ia : String)</li><li>+ LeerNo () : String</li><li>+ EscribirNo (no : String)</li><li>+ LeerU () : String</li><li>+ EscribirU (u : String)</li><li>+ LeerVe () : String</li><li>+ EscribirVe (ve : String)</li><li>+ LeerX () : String</li><li>+ EscribirX (x : String)</li><li>+ LeerY () : String</li><li>+ EscribirY (y : String)</li><li>+ LeerZ () : String</li><li>+ EscribirZ (z : String)</li><li>+ LeerOrden () : String</li><li>+ EscribirOrden (orden : String)</li><li>+ LeerEstado () : int</li><li>+ EscribirEstado (estado : int)</li><li>+ LeerLista () : LinkedList &lt;String&gt;</li><li>+ EscribirLista (ListaPuntos : LinkedList &lt;String&gt;)</li><li>+ LeerOp () : String</li><li>+ EscribirOp (operacion : String)</li><li>+ Convertirla (ia : String) : Integer</li><li>+ ConvertirU (u : String) : Integer</li><li>+ ConvertirOp (operacion : String) : char</li><li>+ ConvertirEg (eg : String) : char</li><li>+ Convertir (Ve : String) : Float</li><li>+ Convertir (dx : String, dy : String, dz : String) : Punto</li></ul>

- LeerFichero:

LeerFichero
- fichero : FileReader
- buffer : BufferedReader
- campo : Campos
+ LeerFichero (fich : String)
+ LeerCampo ( ) : Campos
+ EscribirCampo (campo : Campos)
+ CoCorrecto (co : String) : boolean
+ laCorrecto (ia : String) : boolean
+ UCorrecto (u : String) : boolean
+ NoCorrecto (no : String) : boolean
+ EjeCorrecto (eg : String) : boolean
+ VeCorrecto (ve : String) : boolean
+ PuntoCorrecto (punto : String) : boolean
+ LeerLinea ( ) : Campos

- EscribirFichero:

EscribirFichero
- df : FileOutputStream
- fichero_salida : DataOutputStream
- fich_sal : FileWriter
- buffer_sal : BufferedWriter
+ EscribirFichero (Nombre : String)
+ Escribir_AOB (Nombre : String)
+ Escribir_EOB (Nombre : String)
+ Escribir_ATR (Nombre : String)
+ Escribir_AAC (ia : String)
+ Escribir_APB (Nombre : String, ia : String)
+ Escribir_DAC (Nombre : String, ia : String)
+ Escribir_EAC (ia : String)
+ Escribir_MDO (ob : Objeto)
+ Escribir_MAO (ob : Objeto)
+ Escribir_CSU (Nombre : String, Sup : float)
+ escribir_caso (error : int)

- GestionTransformaciones:

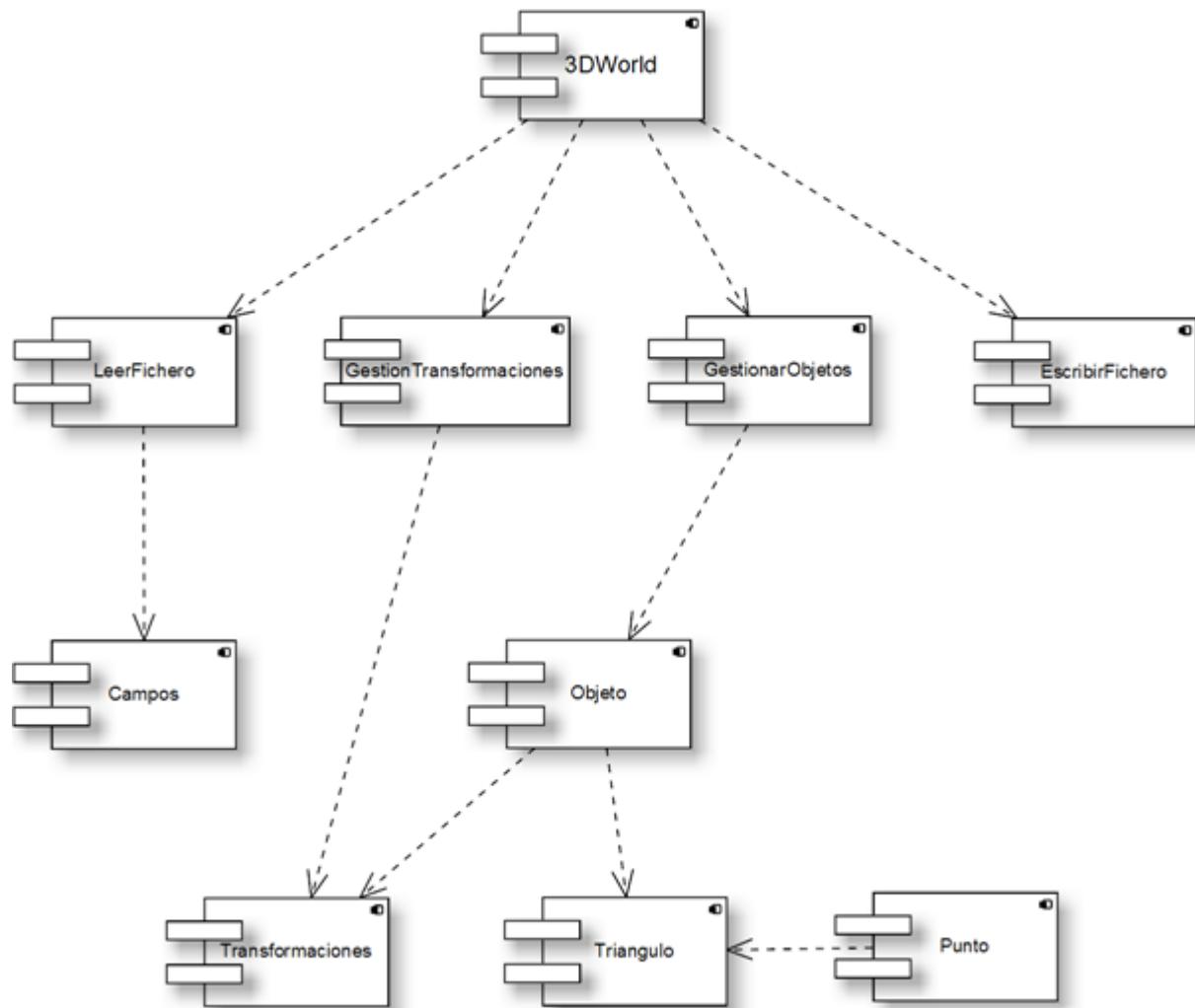
GestionTransformaciones
- ListaTransformaciones : LinkedList <Transformaciones>
+ getListaTransformaciones ( ) : LinkedList <Transformaciones>
+ setListaTransformaciones (ListaTransformaciones : LinkedList <Transformaciones>)
+ Esta (ia : int) : boolean
+ DevolverTrans (ia : int) : Transformaciones
+ AgregarAccion (trans : Transformaciones)
+ EliminarAccion (identificador : int)

- GestionarObjetos:

GestionarObjetos
- ListaObjetos : LinkedList <Objeto>
+ GestionarObjetos ( )
+ getListaObjetos ( ) : LinkedList <Objeto>
+ setListaObjetos (ListaObjetos : LinkedList <Objeto>)
+ Esta (Nombre : String) : boolean
+ Identificador (Nombre : String) : Integer
+ AgregarObjeto (Nombre : String, Color : String)
+ EliminarObjeto (Nombre : String)
+ AgregarTriangulo (Punto1 : Punto, Punto2 : Punto, Punto3 : Punto, Nombre : String)
- superficieTriangulo (tri : Triangulo) : float
+ SuperficieObjeto (Nombre : String) : float
+ AplicarAccionTransformacion (Nombre : String, trans : Transformaciones)
+ MostrarDatosObjeto (Nombre : String) : Objeto
+ DeshacerAccionTransformacion (Nombre : String)
+ EstaAplicada (ia : int) : boolean
+ Estala (ia : int, ob : Objeto) : boolean
+ TieneTriangulos (Nombre : String) : boolean

### 3. IMPLEMENTACIÓN

#### 3.1. Diagrama de componentes:



#### 3.2. Codificación en el lenguaje elegido:

El lenguaje que se ha empleado para codificar el programa es “JAVA” y todo el código fuente se encuentra dentro de la carpeta con nombre “Código” dentro del CD entregado.

**modelo**

**ORIENTADO AL**

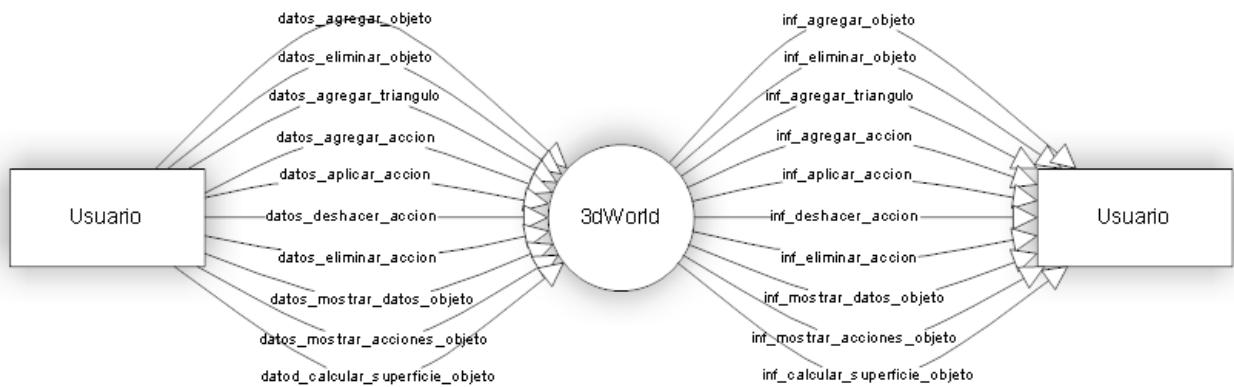
**FLUJO DE DATOS**

## 1. ANÁLISIS

### 1.1. Modelo ambiental: diagrama de contexto y lista de sucesos.

En este apartado se incluye el diagrama de contexto el cual nos define las interfaces entre el sistema y el usuario, así como nos indica que cosas forman parte del sistema.

El diagrama de contexto correspondiente a nuestra práctica es el siguiente:



Como podemos observar el usuario es el encargado de proporcionar los datos a través de un fichero, posteriormente el programa 3dWorld es el encargado de procesar dichos datos, escribiendo en otro fichero la salida correspondiente.

Nuestra lista de sucesos es la siguiente:

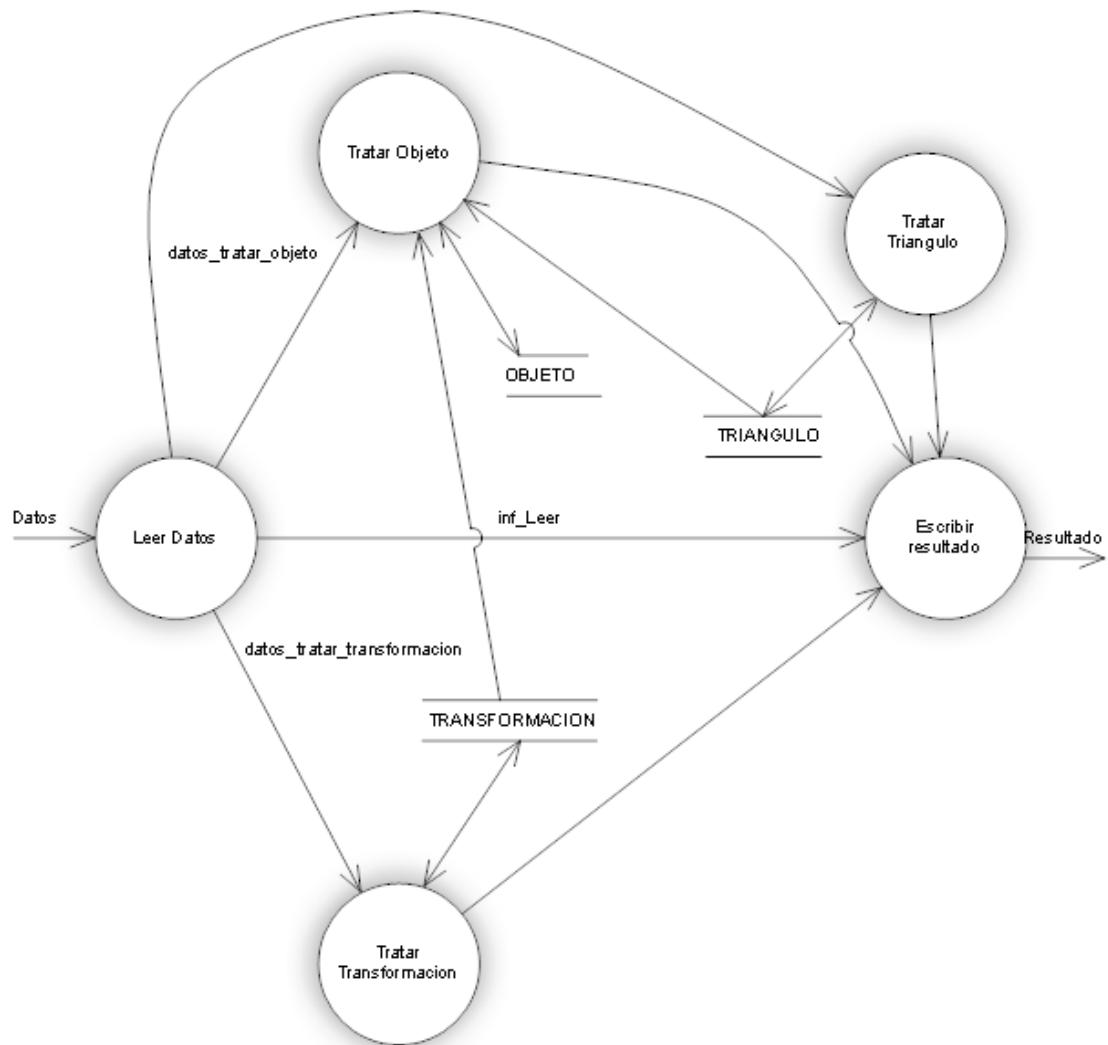
- Agregar Objeto
- Eliminar Objeto
- Agregar Triángulo
- Agregar Acción
- Aplicar Acción
- Deshacer Acción
- Eliminar Acción
- Mostrar Datos Objeto
- Mostrar Acciones Objeto
- Calcular Superficie Objeto

## 1.2. Modelo de comportamiento.

### 1.2.1. Modelo de proceso.

En primer lugar mostramos todos los DFD

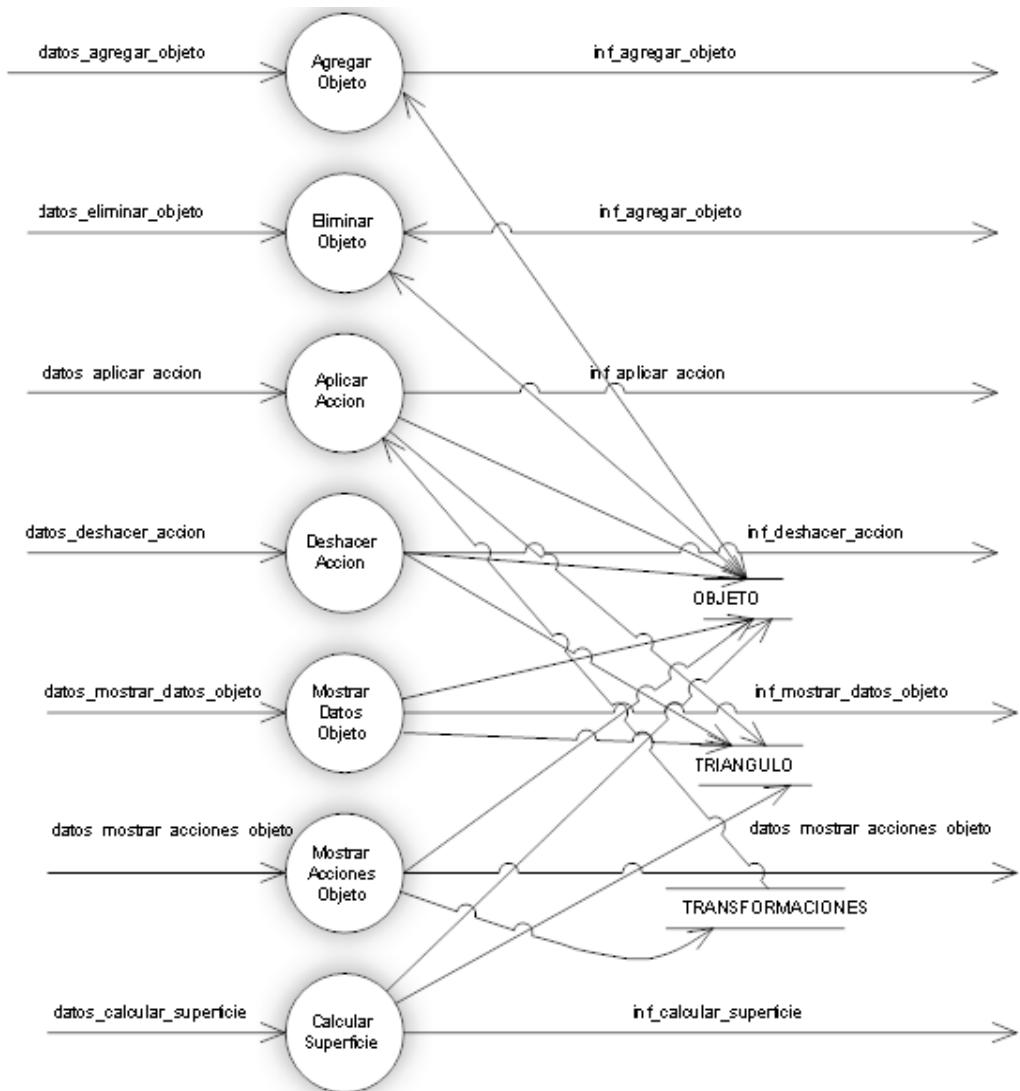
- Diagrama de flujo de datos del nivel 1:



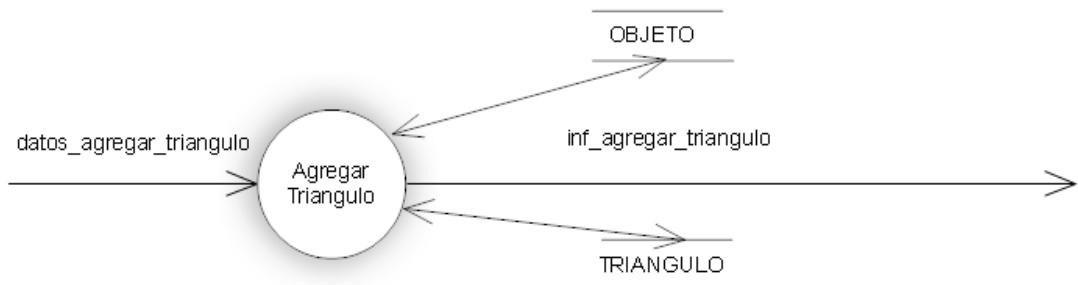
El funcionamiento es el siguiente: Leer datos se encarga de obtener los datos y comprobar que el formato y el valor de los mismos es el correcto, si no fuera así pasa directamente a Escribir resultado donde se escribiría el error correspondiente. Si los datos fueran correctos, pasaríamos a Tratar Transformación, Tratar Triángulo ó Tratar Objeto, dependiendo de la operación, estos procesos operan con los almacenes correspondientes y por último pasaríamos a Escribir Resultado para la escritura de los resultados.

■ Diagrama de flujo de datos de nivel 2:

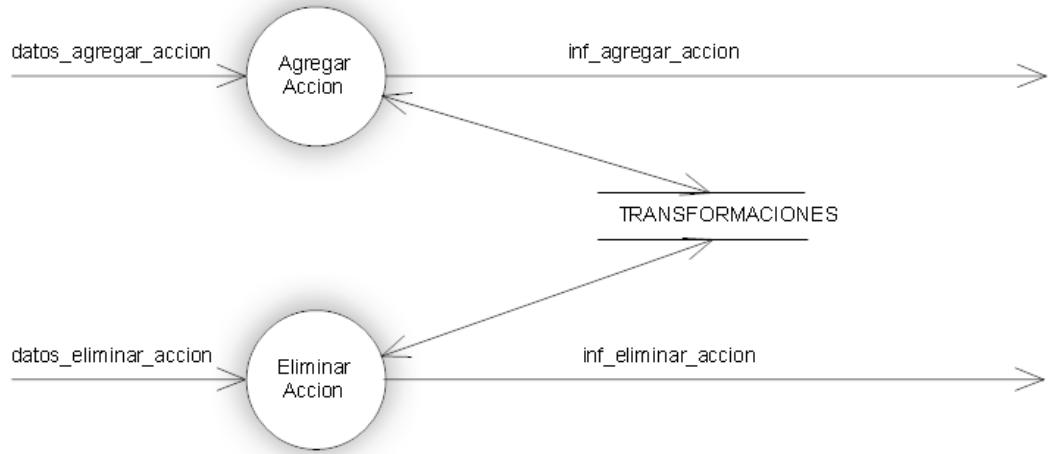
**Diagrama de Tratar Objeto:**



**Diagrama de Tratar Triángulo:**



### Diagrama de Tratar Transformación:



- Especificación de 3 procesos primitivos:

#### 1. Agregar Objeto

**Entrada:** datos\_agregar\_objeto // no + co  
Objeto (Almacen)

#### Proceso:

```

If esta_no(no, lista_objetos)
  Then
    Informacion_agregar_objeto = "ERROR: objeto ya existe"
  Else
    Agregar_objeto (no, co)
  End if
end Proceso
  
```

**Salida:** inf\_agregar\_objeto  
Objeto (Almacén)

#### 2. Eliminar Acción

**Entrada:** datos\_eliminar\_accion // ia  
Objeto (Almacen)  
Transformaciones(Almacen)

**Proceso:**

```
If !(esta_ia(ia, lista_transformaciones))
Then
    Informacion_eliminar_accion = "ERROR:la acción no existe"
Else
    If esta_ia(ia, lista_transformaciones_objeto)
    Then
        Informacion_eliminar_accion = "ERROR:la acción esta
        aplicada"
    Else
        Eliminar_Accion (ia)
    End If
End if
end Proceso

Salida: inf_eliminar_accion
Objeto (Almacén)
Transformaciones(Almacen)
```

**3. Calcular Superficie Objeto**

**Entrada:** datos\_calcular\_superficie // no  
Objeto (Almacen)

**Proceso:**

```
If !esta_no(no, lista_objetos)
Then
    Informacion_calcular_superficie_objeto = "ERROR: El objeto no
    existe"
Else
    If !hay_triangulos(no,Lista_Objetos)
    Then
        Informacion_calcular_superficie_objeto = "ERROR: El
        objeto no tiene triangulos"
    Else
        Calcular_Superfiie_Objeto (no)
    End If
End if
end Proceso

Salida: inf_calcular_superficie_objeto
Objeto (Almacén)
```

- Diccionario de datos completo:

Datos = tipo\_operacion + parámetros\_operacion

Tipo\_operacion = [ 21 \* Agregar objeto | 22 \* Eliminar objeto| 23 \* Agregar Triangulo | 31\*Agregar accion| 32 \* Aplicar accion | 33 \* Deshacer accion | 34 \* Eliminar accion | 41 \* Mostrar datos de objeto |42 \* Mostrar acciones de objeto|43 \* Calcular superficie de objeto]

Parametros\_operacion = [datos\_AgregarObjeto | datos\_EliminarObjeto | datos\_AgregarTriangulo | datos\_AgregarAccion | datos\_AplicarAccion | datos\_DeshacerAccion | datos\_EliminarAccion | datos\_MostrarDatosObjeto | datos\_MostrarAccionesObjeto | datos\_CalcularSuperficieObjeto ]

Datos\_AgregarObjeto = no+co

Datos\_EliminarObjeto = no

Datos\_AgregarTriangulo = x1+y1+z1+x2+y2+z2+x3+y3+z3+no

En caso de AgregarAccion tendríamos tres casos:

Datos\_AgregarAccion = ia+'t'+dx+dy+dz

Datos\_AgregarAccion = ia+'g'+eg+u

Datos\_AgregarAccion = ia+'e'+ve

Datos\_AplicarAccion = no+ia

Datos\_DeshacerAccion = no

Datos\_EliminarAccion = ia

Datos\_MostrarDatosObjeto = no

Datos\_MostrarAccionesObjeto=no

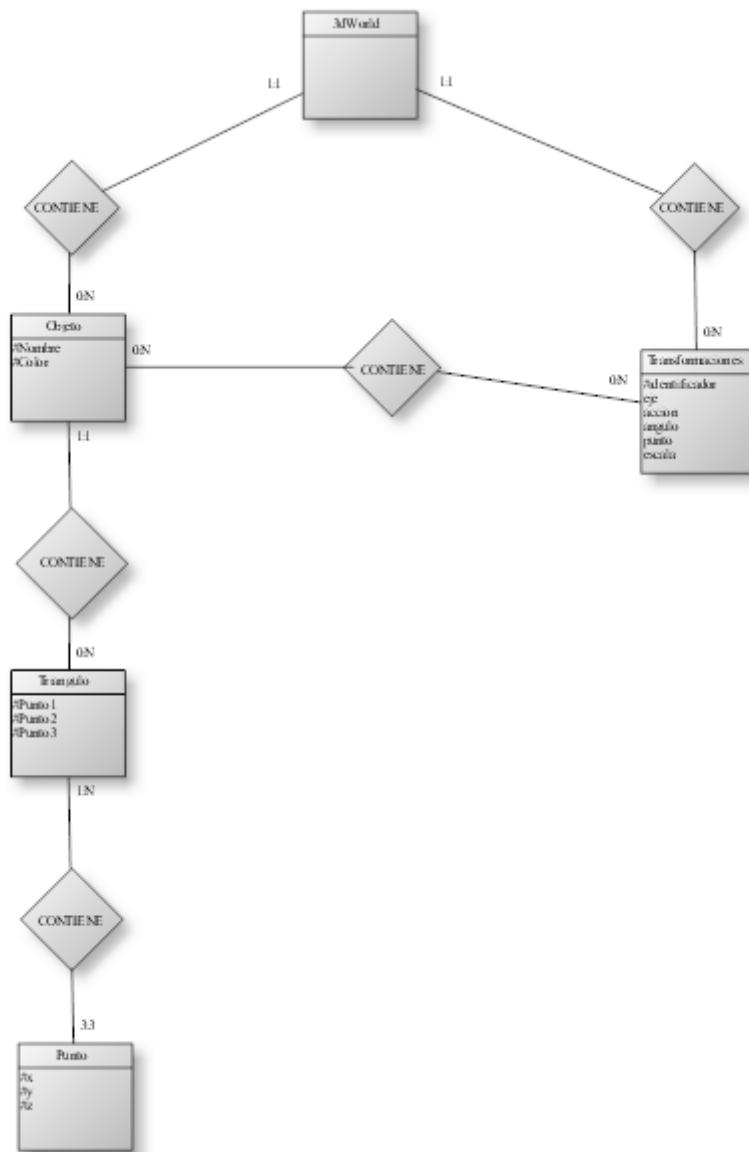
Datos\_CalcularSuperficieObjeto=no

Donde tenemos las siguientes abreviaturas:

Abreviatura	Significado	tipo
co	Color del objeto	string
dx	Desplazamiento en un eje	Punto
dy		
dz		
eg	Eje de giro	char
ia	Identificador de acción	int
no	Nombre del objeto	string
u	Angulo de giro	int
ve	Vector de escala	int
x	Coordenada tridimensional	Punto
y		
z		

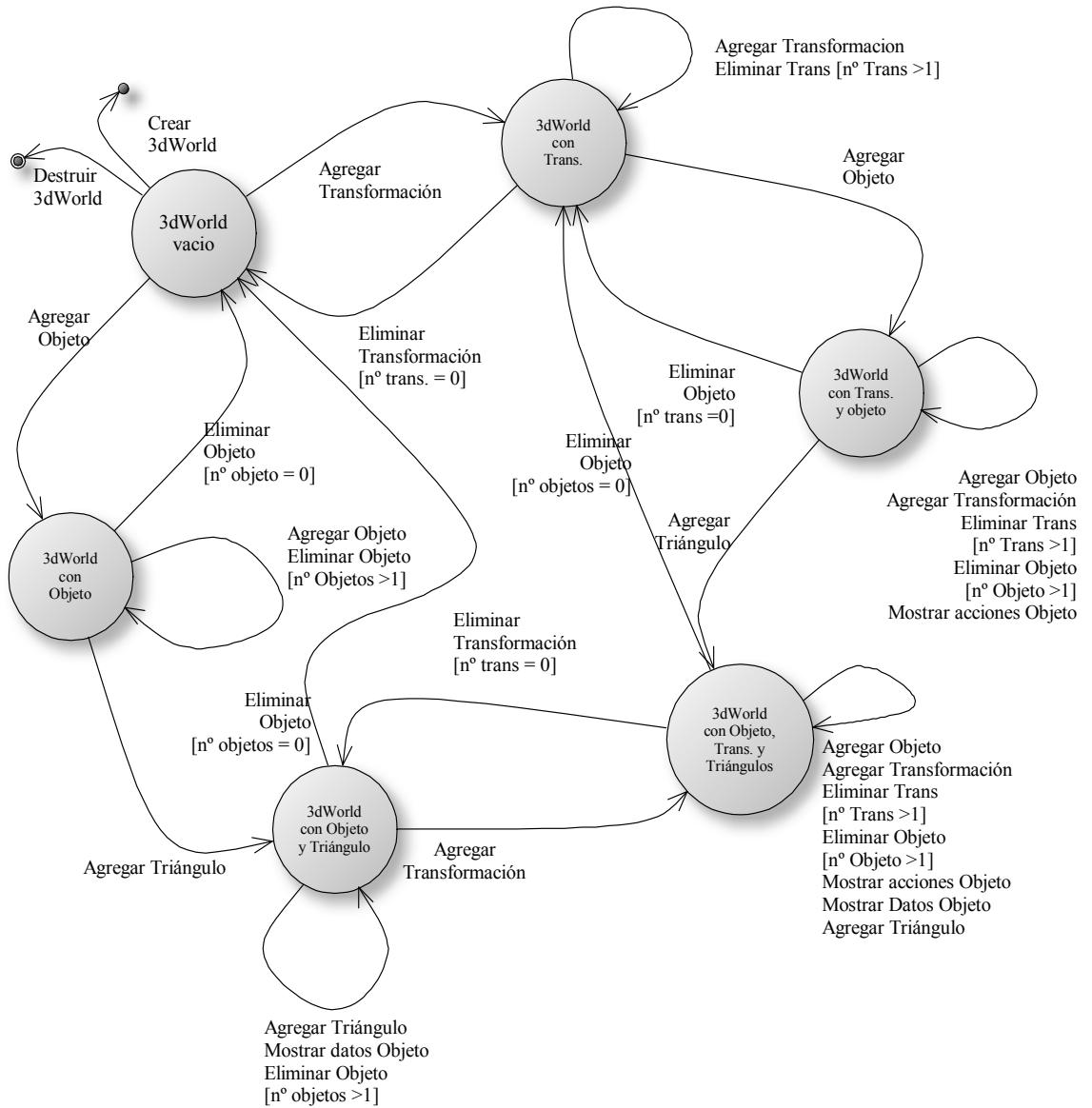
### 1.2.2. Modelo de datos: diagrama E/R.

A continuación incluimos el diagrama de entidad relación con todos sus atributos y la relación existente entre entidades:



### 1.2.3. Diagrama de transición entre estados.

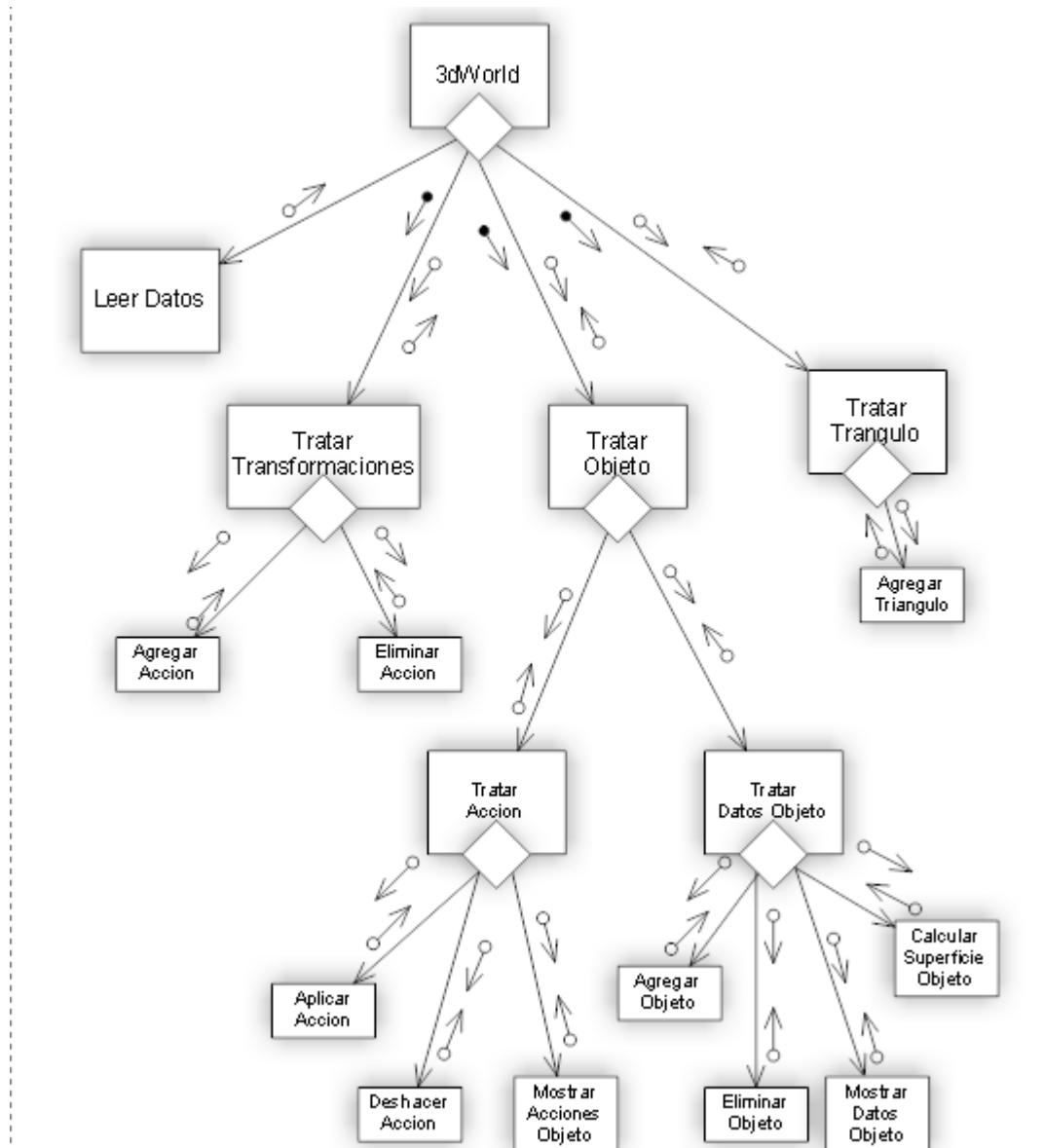
Mediante este diagrama podemos comprobar el comportamiento del programa en tiempo real. Las transiciones entre estados van en función de los eventos que se reciban.



## 2. DISEÑO

### 2.1. Modelo de implantación de programas.

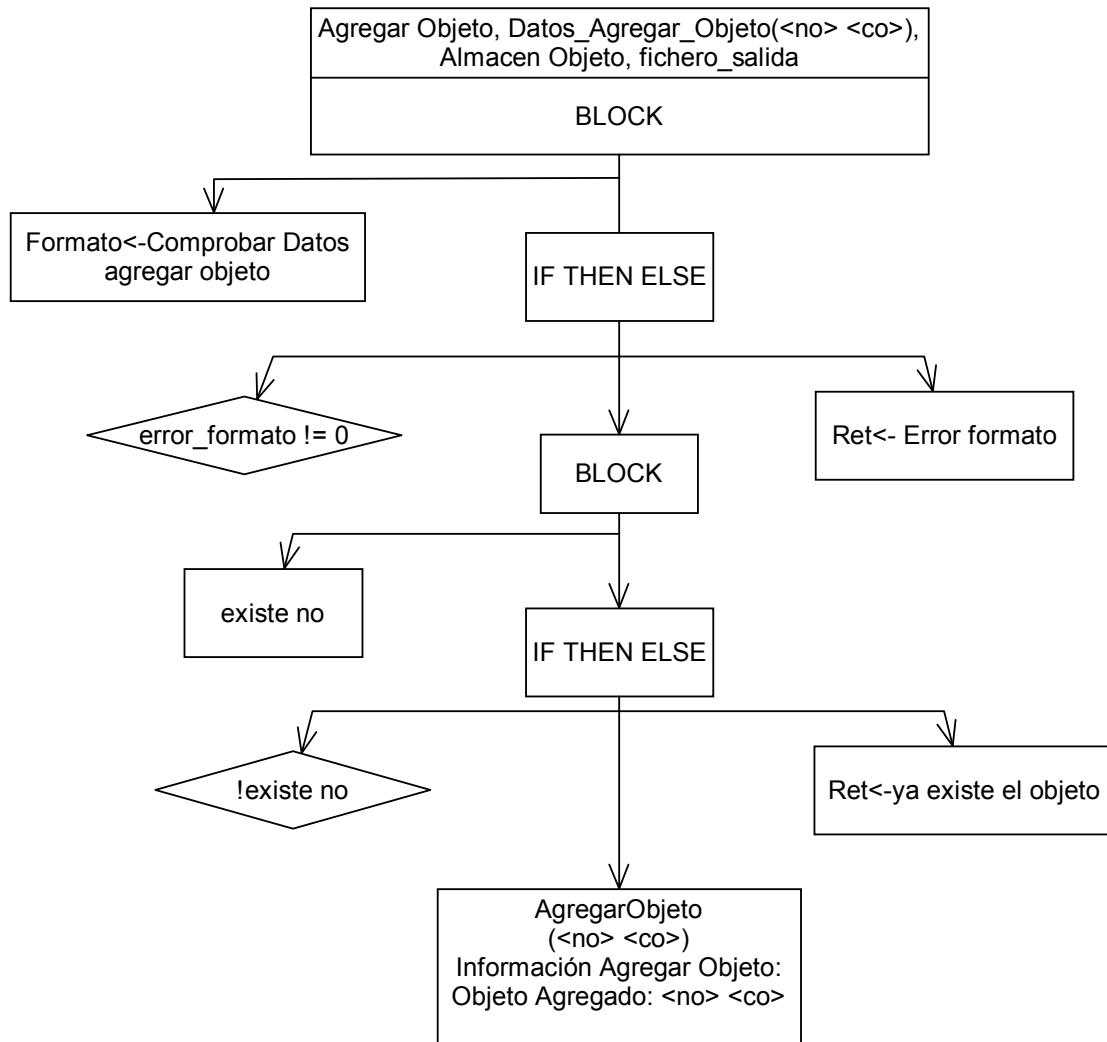
#### 2.1.1. Diagrama de estructura refinado.



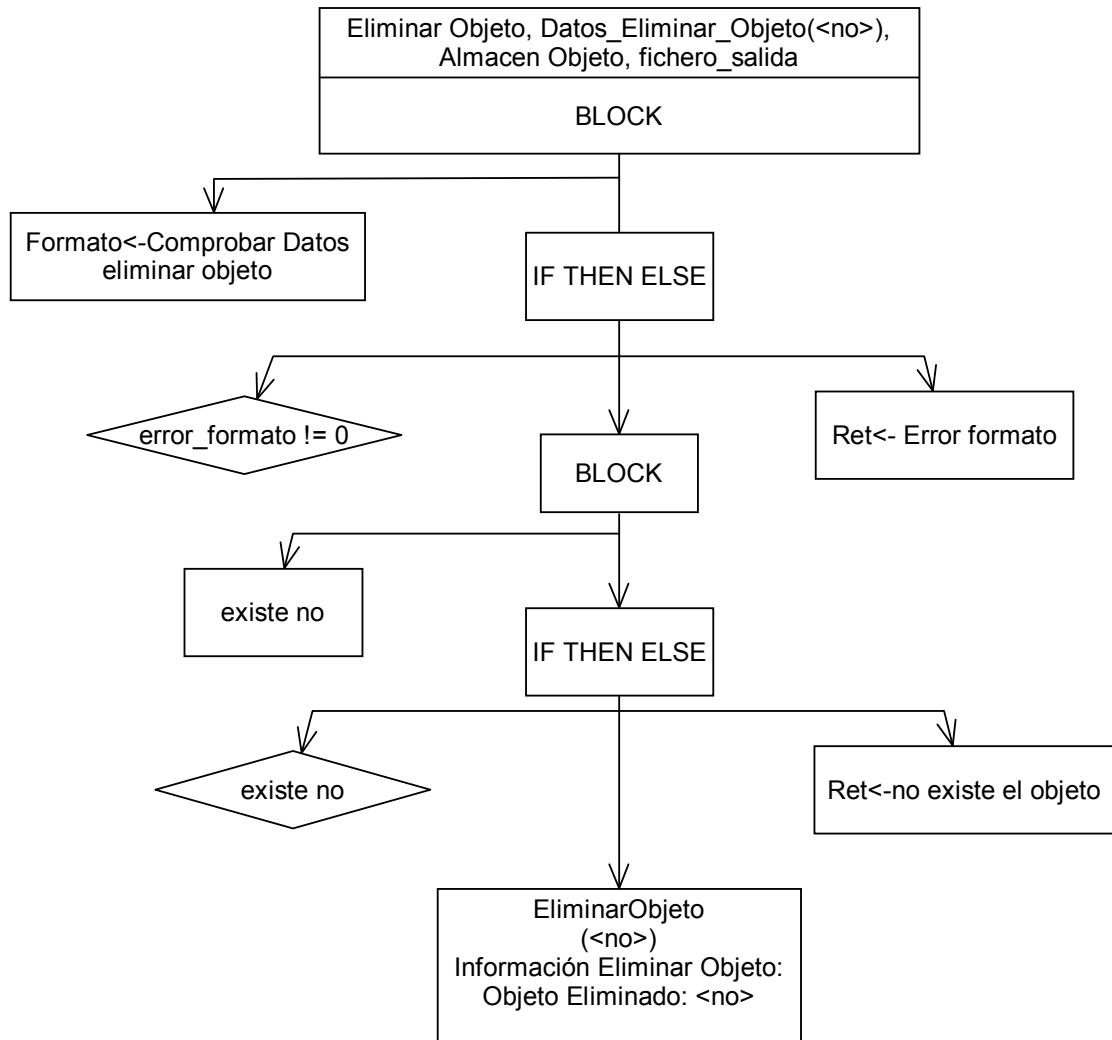
Para realizar el refinamiento debido a que el módulo Tratar Objeto tenía un fan-out de 6 por lo que hemos creado dos submódulos (Tratar Acción y Tratar Datos Objeto) para reducir el fan-out, por otro lado hemos aumentado el fan-in del modulo escribir resultados y una vez aplicada la reducción de cohesión lógica se obtiene el diagrama anteriormente mostrado.

### 2.1.2. Diagrama de estructura de todos los módulos.

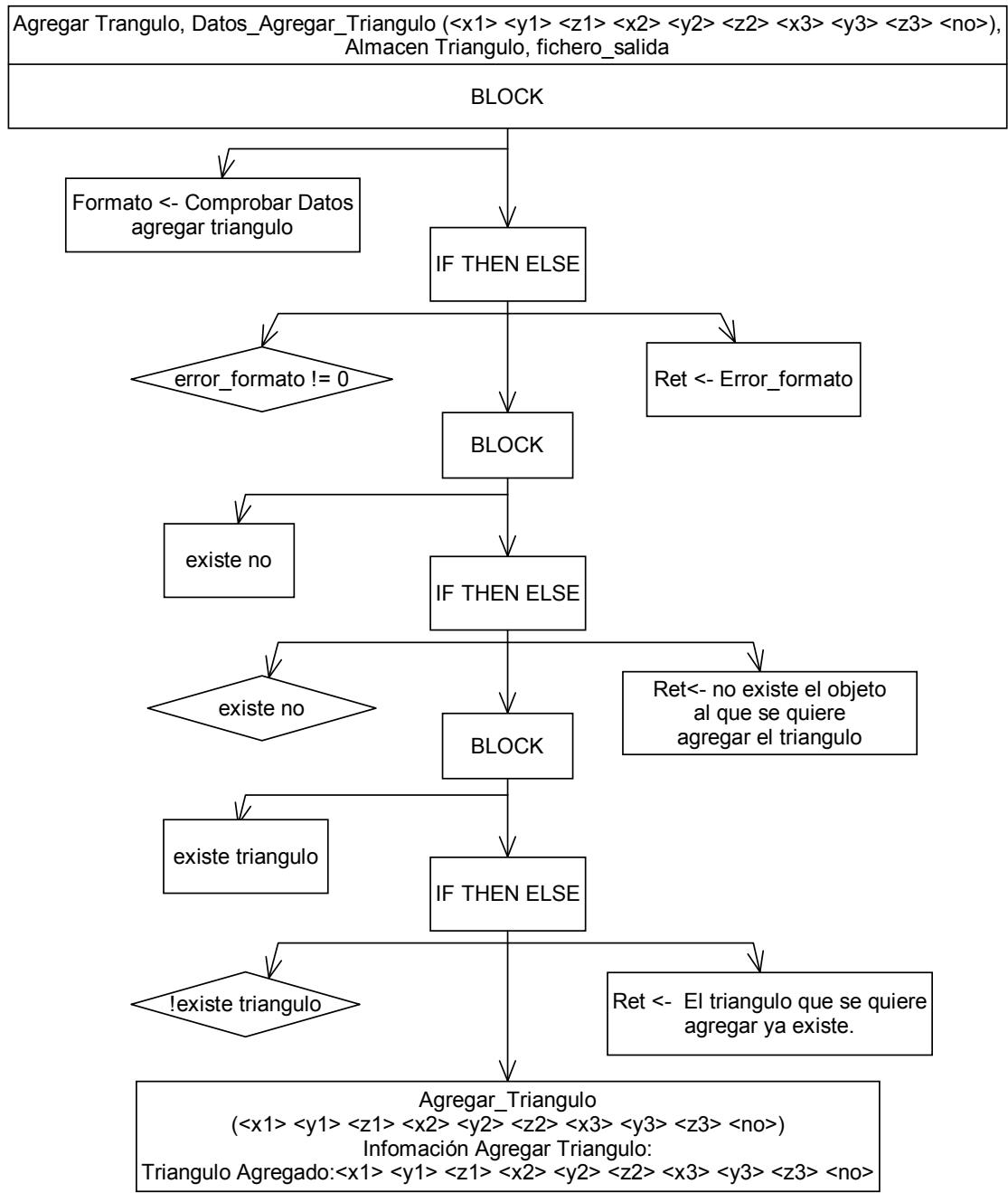
#### - Agregar Objeto



- Eliminar Objeto

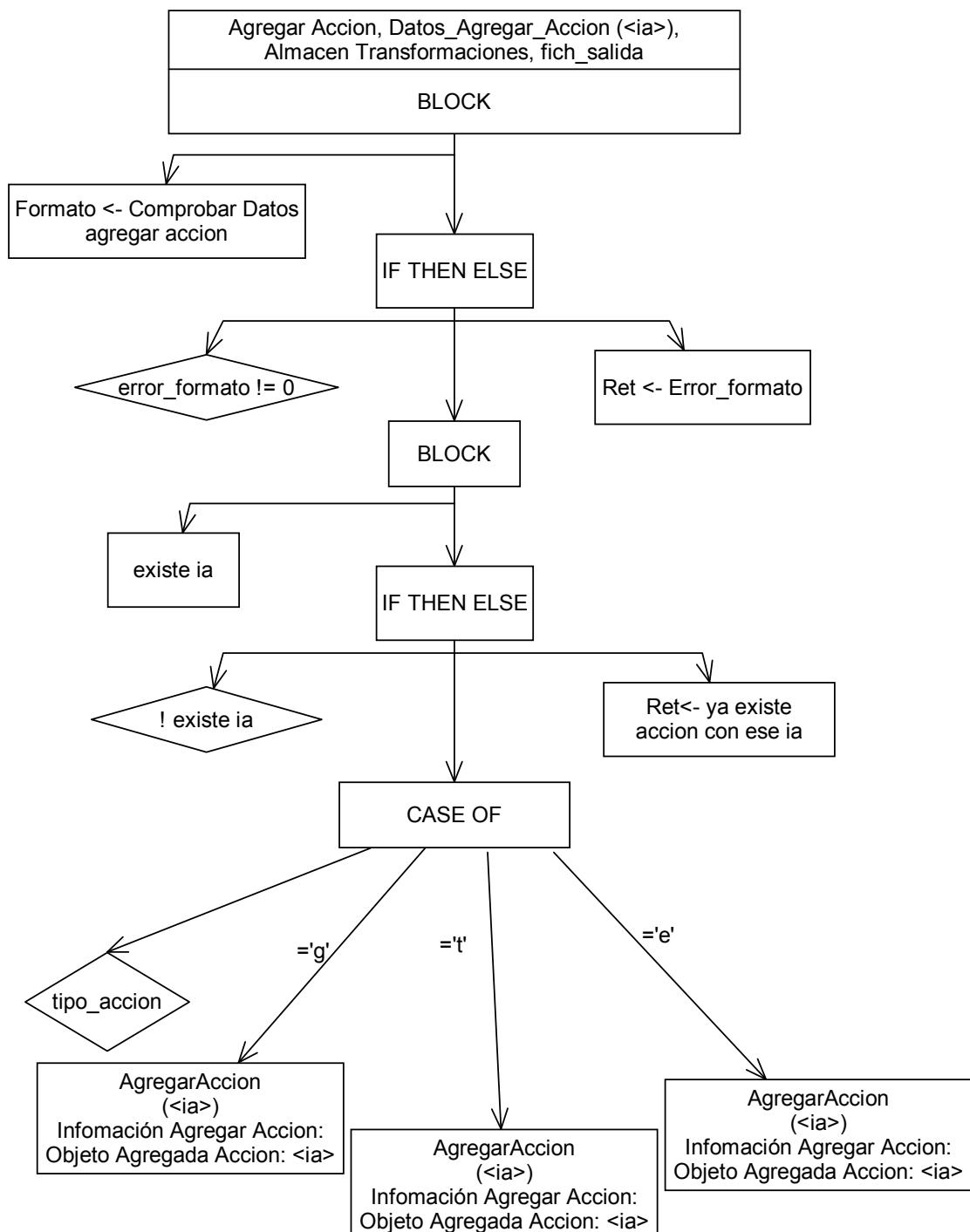


- Agregar Triángulo

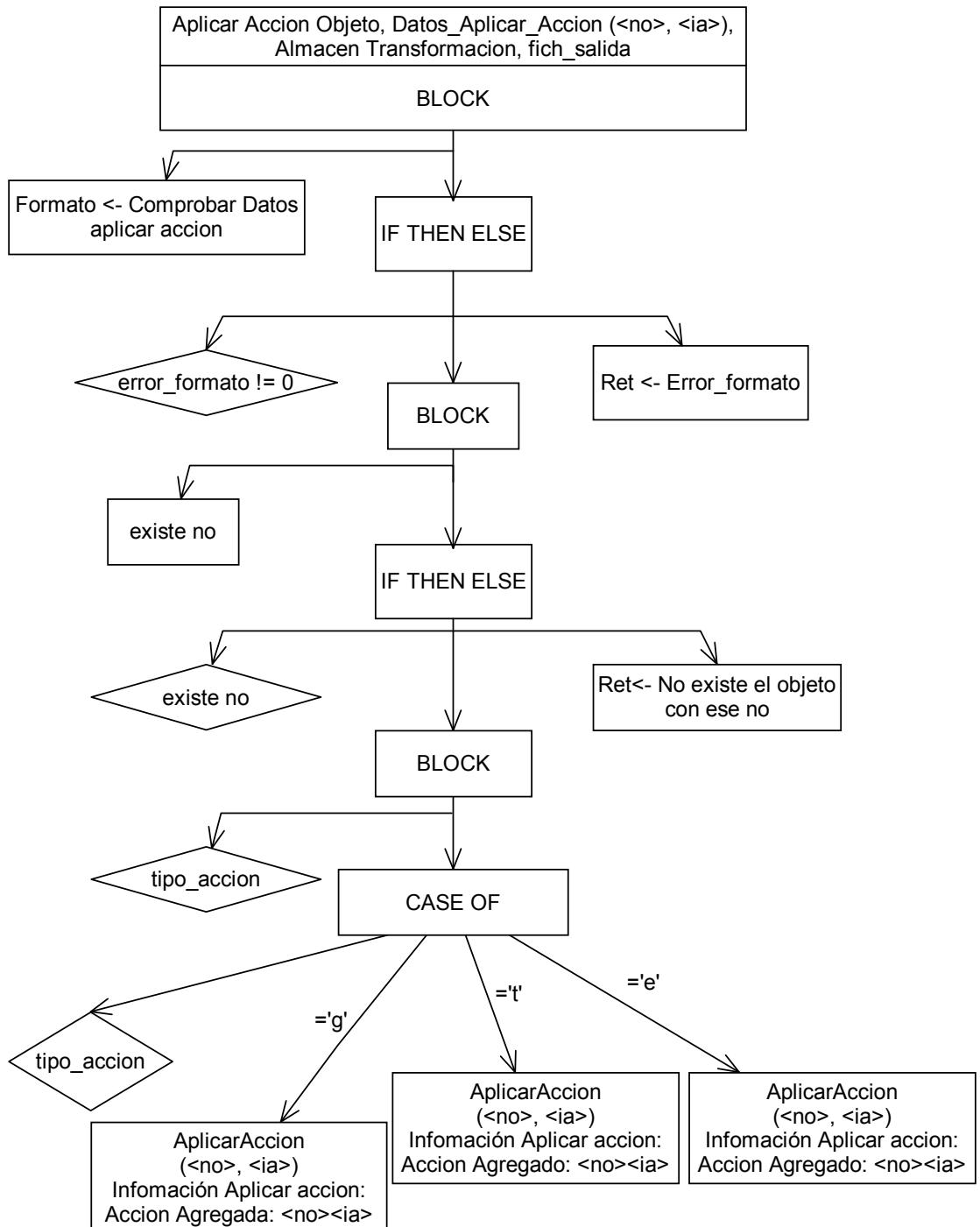


- Agregar acción

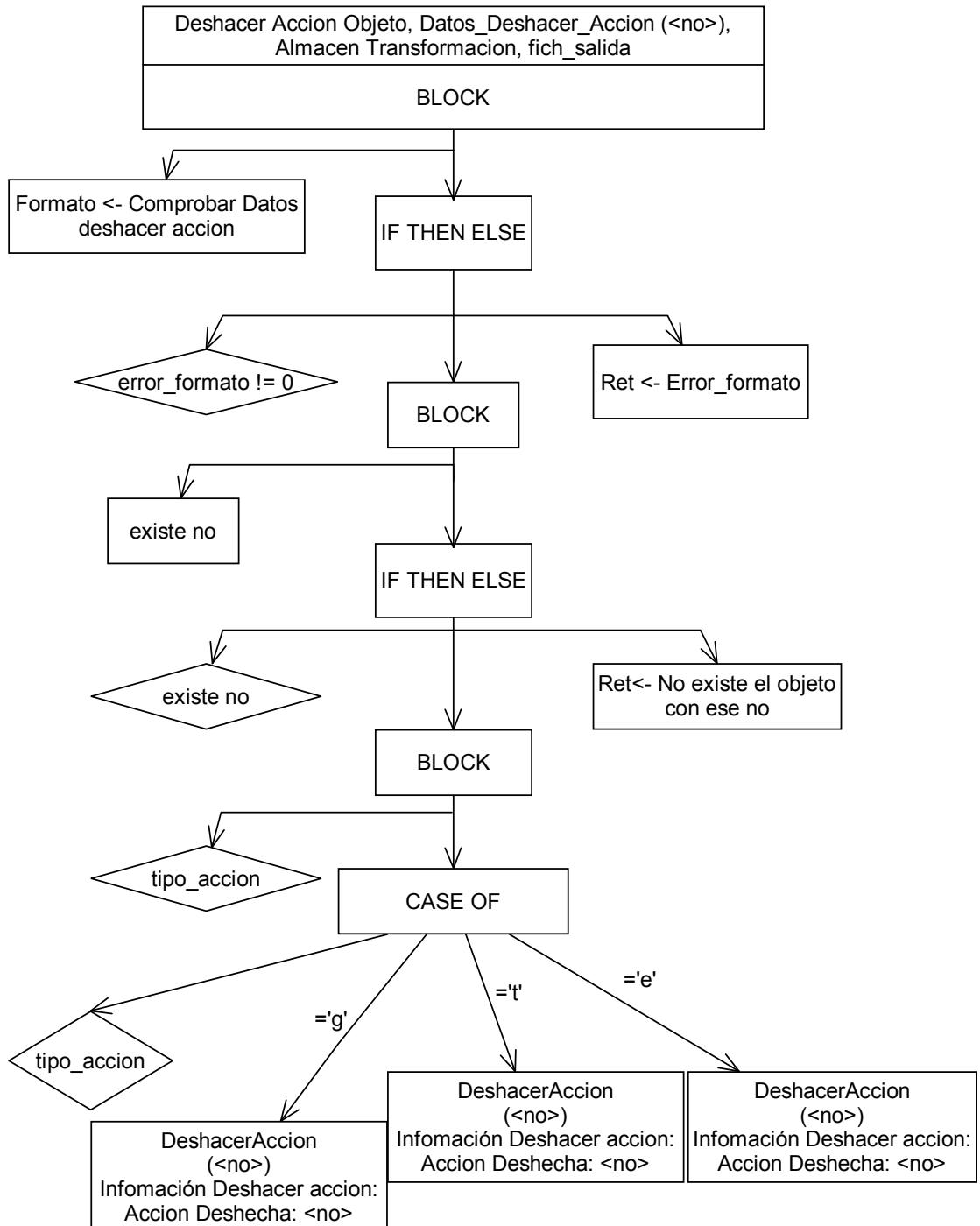
Debido a las 3 opciones a la hora de agregar una acción para mayor comprensión este diagrama será independiente del tipo de transformación que sea.



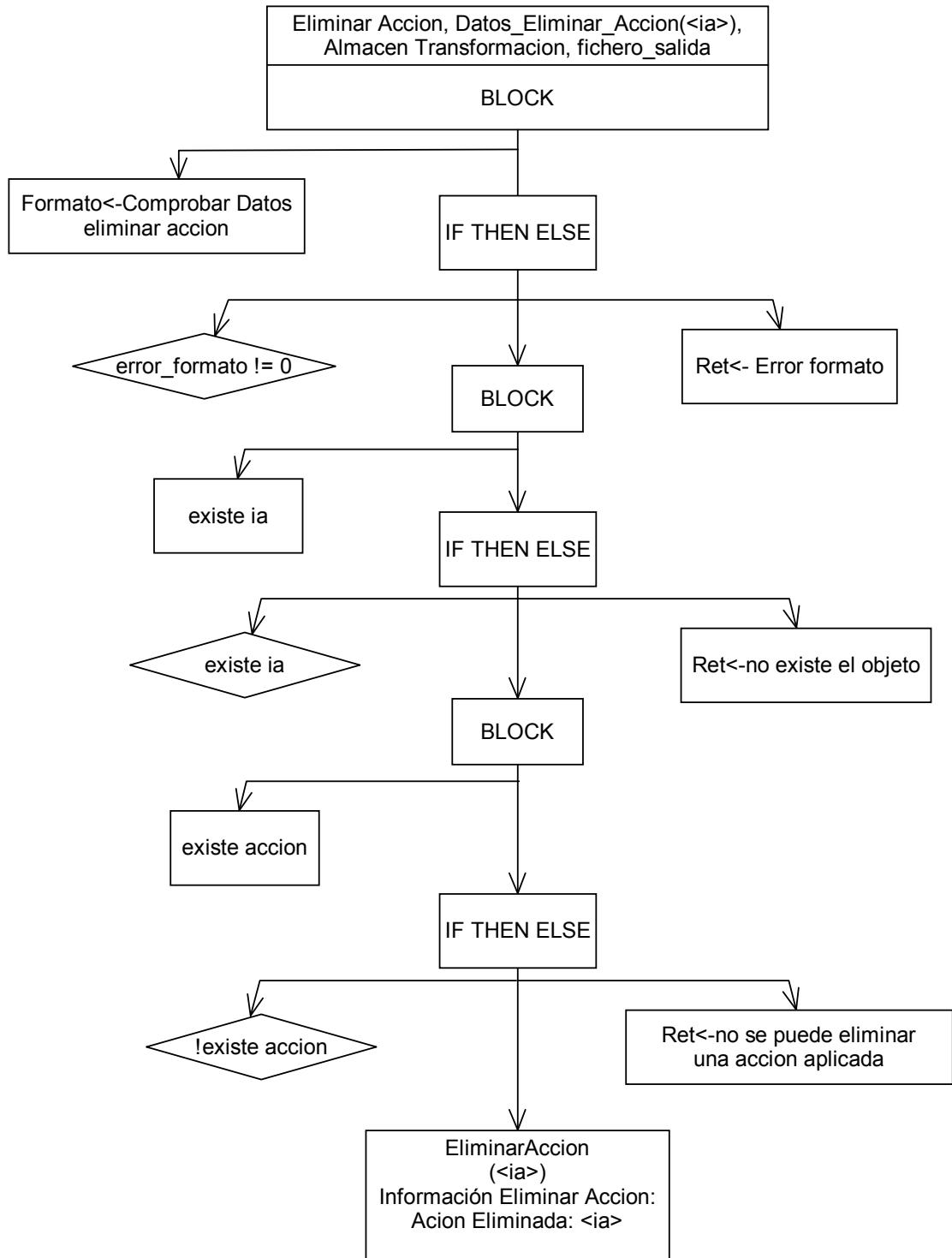
- Aplicar acción



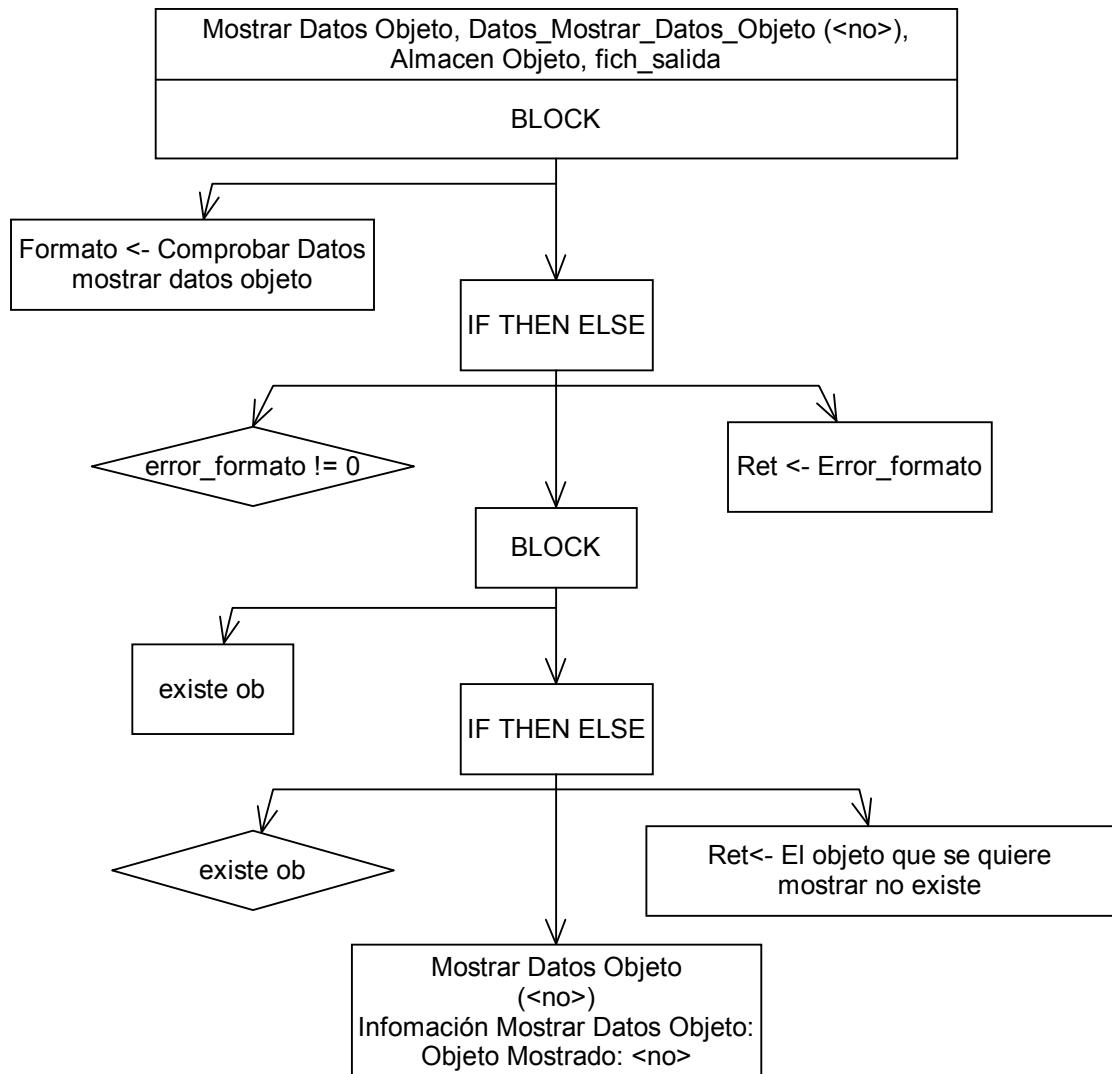
- Deshacer acción



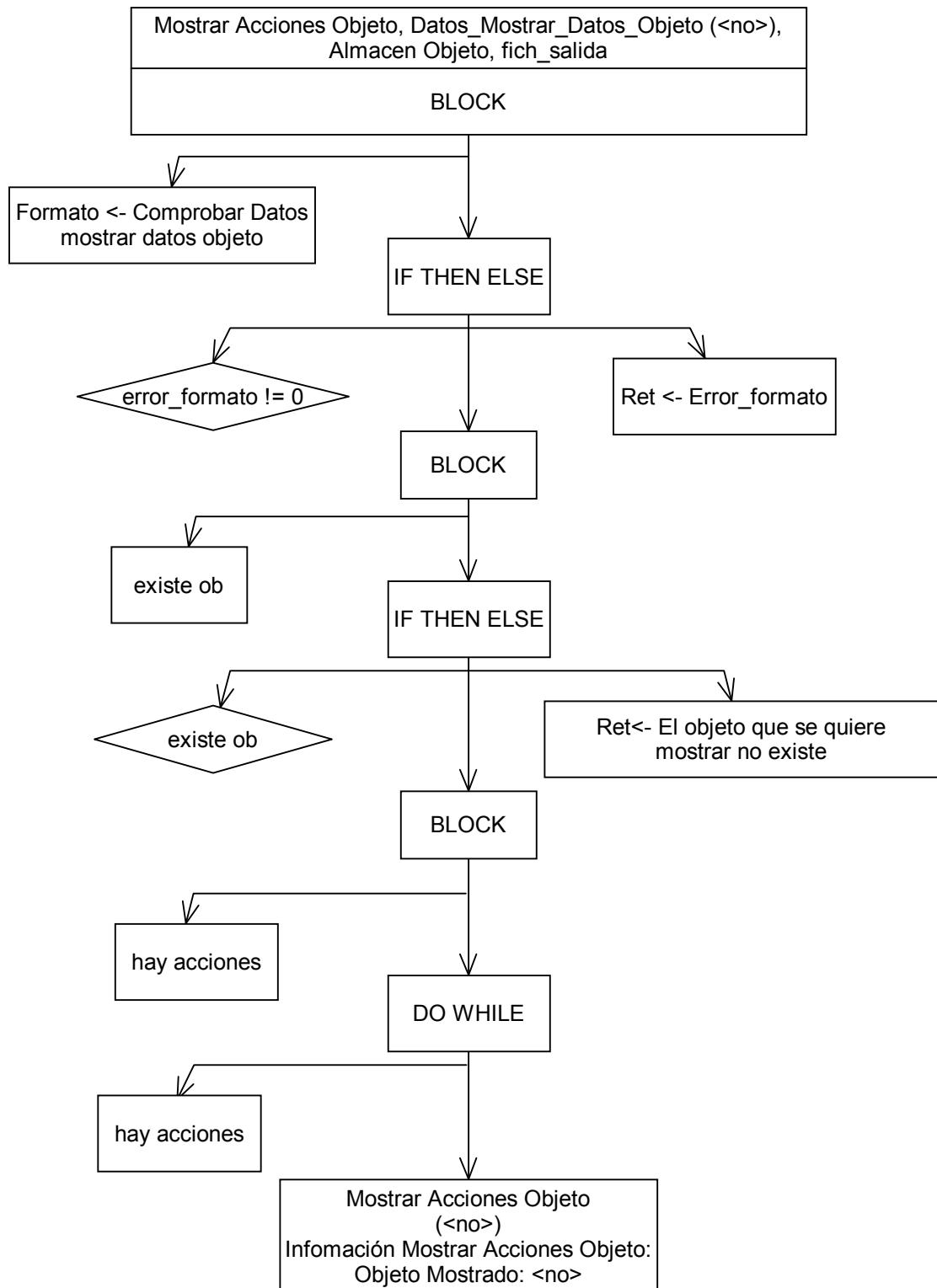
- Eliminar acción



- Mostrar datos Objeto



- Mostrar Acciones Objeto



- Calcular Superficie Objeto

