Anexo F. Relación de los métodos y funciones de CrisolScript.

Índice.

F.1. El objeto Game.

El objeto `Game` se encargara de representar a *Crisol* en calidad de apliación. Esto quiere decir que trabajará principalmente con cuadros de diálogo e interfaces. A continuación se lista la definición de los métodos que poseerá y su explicación.

```
void Game.QuitGame(void)
```

*Descripción***.**

- Quita la partida actual de forma inmediata, regresando al menú principal.

Notas.

- Sólo se podrá lanzar desde el estado de interfaz principal o desde el inventario del jugador.

```
void Game.WriteToConsole(string Text)
```

*Descripción***.**

- Escribe en la consola de juego el texto pasado por parámetro. La consola de juego puede ser la que se halle en el menú de interfaz principal o en la pantalla de inventario. Se usará en la que se halle el usuario.

Parámetros.

- `Text`. Texto a escribir en la consola de juego.

```
number Game.IsKeyPressed(number Key)
```

*Descripción***.**

- Comprueba si el identificador de tecla recibido se halla, o no, pulsado.

Parámetros.

- `Key`. Identificador de la tecla a consultar.

Devolución.

- Si la tecla se halla pulsada 1, en caso contrario 0.

```
void Game.ActiveAdviceDialog(string szText)
```

*Descripción***.**

- Activa el cuadro de diálogo destinado a mostrar un texto con un aviso al usuario.

Parámetros.

- `szText`. Texto de aviso.

Devolución.

- Si todo ha ido bien `true`, en caso contrario `false`.

```
number Game.ActiveQuestionDialog(string Text,
                                 number UseCancelOption)
```

*Descripción***.**

- Activa el cuadro de diálogo destinado a realizar una pregunta al usuario y obtener una respuesta de éste..

Parámetros.

- `Text`. Texto con la pregunta a realizar al usuario.
- `UseCancelOption`. Si es un valor superior o igual a 1 se habilitará la opción de cancelación y si es un valor interior a 1 no.

Devolución.

- Si el usuario pulsó la opción de aceptación un valor de 1, si pulsó la opción de rechazo, un valor de 2. En caso de pulsar la opción de cancelación un valor de 0.

```
void Game.ActiveTextReaderDialog(string Title,
                                 string FileName)
```

*Descripción***.**

- Activa el cuadro de diálogo destinado a mostrar un archivo de texto.

Parámetros.

- `Title`. Título asociado al cuadro de diálogo.
- `FileName`. Nombre del archivo de texto con los datos a mostrar.

```
number Game.AddOptionToTextSelectorDialog (number IDOption,
                                           string Text,
                                           number CanSelect)
```

*Descripción***.**
- Permitirá añadir opciones al cuadro de diálogo dedicado a mostrar texto seleccionable.

Parámetros.
- `IDOption`. Identificador de la opción. No podrá valer 0.
- `Text`. Texto asociado a la opción.
- `CanSelect`. Si su valor es igual o superior a 1, la opción se podrá seleccionar. Si vale menos de 1 no.

Devolución.
- Si la opción pudo ser registrado 1, en caso contrario 0.

```
void Game.ResetOptionsInTextSelectorDialog(void)
```

*Descripción***.**
- Reseteará todas las opciones que hayan asociadas.

```
number Game.ActiveTextSelectorDialog(string Title,
                                     number UseCancelOption)
```

*Descripción***.**
- Activará el cuadro de diálogo de selección de texto con las opciones que se hayan añadido. Si no hubiera ninguna, la activación no será posible.

Parámetros.
- `Title`. Texto con el título asociado al cuadro de diálogo.
- `UseCancelOption`. Si es un valor superior o igual a 1 se habilitará la opción de cancelación y si es un valor inferior a 1 no.

Devolución.
- El cuadro de diálogo esperará por la selección, por parte del usuario, de una de las opciones que fueron registradas, retornando su identificador. Si el valor que retorna es de cero, significará que se escogió la opción de cancelación.

```
void Game.PlayMIDIMusic(string FileName,
                        number Loop)
```

*Descripción***.**
- Reproducirá un fichero musical MIDI. Si ya había otro siendo reproducido, quedará sustituido por el aquí especificado.

Parámetros.
- `FileName`. Nombre del fichero musical MIDI a reproducir. Se deberá de incluir con su extensión.
- `Loop`. Si es un valor superior o igual a 1 la reproducción será cíclica y si es un valor inferior a 1 sólo se reproducirá una sola vez.

```
void Game.StopMIDIMusic(void)
```

*Descripción***.**
- Detendrá el fichero musical MIDI que se halle en reproducción.

```
void Game.PlayWAVAmbientSound(string FileName)
```

*Descripción***.**
- Reproducirá un fichero musical WAV como sonido ambiente del universo de juego.

Parámetros.
- `FileName`. Nombre del fichero WAV a reproducir. Se deberá de incluir con su extensión.

```
void Game.StopWAVAmbientSound(void)
```

*Descripción***.**
- Detendrá el fichero musical WAV que se halle en reproducción.

```
void Game.ActiveTradeItemsInterfaz(entity EntityToTrade)
```

*Descripción***.**
- Activará el interfaz de intercambio de ítems entre el jugador y otra entidad, que será la recibida por parámetro y deberá de ser contenedora.

Parámetros.
- `EntityToTrade`. Entidad con la que el jugador iniciará el intercambio de ítems.

Notas.

- Sólo se podrá lanzar desde el estado de interfaz principal.

```
void Game.ActiveConversatorInterfaz(entity EntityToConverse)
```

*Descripción***.**

- Activará el interfaz de conversación entre el jugador y una entidad que será recibida por parámetro. Para poder añadir opciones y comenzar a utilizar el interfaz de conversación, siempre se deberá de activar el mismo mediante este método.

Parámetros.

- `EntityToConverse`. Entidad con la que el jugador iniciará la conversación.

Notas.

- Sólo se podrá lanzar desde el estado de interfaz principal.

```
number Game.AddOptionToConversatorInterfaz(number IDOption,
                                           String OptionText)
```

*Descripción***.**

- Añadirá opciones de conversación a la interfaz.

Parámetros.

- `IDOption`. Identificador de la opción escogida. No se podrá pasar identificadores de valor inferior a 1.
- `OptionText`. Texto asociado a dicho identificador, que será el que se muestre en pantalla.

Devolución.

- Si se pudo añadir la opción 1, en caso contrario 0.

Notas.

- Sólo se podrá lanzar si el interfaz de conversación se halla activo.

```
void Game.ResetOptionsInConversatorInterfaz(void)
```

*Descripción***.**

- Reseteará todas las opciones que se hayan añadido hasta el momento.

Notas.

- Sólo se podrá lanzar si el interfaz de conversación se halla activo.

```
number Game.GetOptionFromConversatorInterfaz(void)
```

*Descripción***.**

- Ordenará mostrar todas las opciones añadidas hasta el momento y preguntar al usuario para que escoja una de ellas. Cuando haya escogido la deseada, el método retornará el identificador que se asoció a la misma.

Devolución.

- El identificador de la opción escogida. Si no se pudo escoger ninguna por finalización externa, un valor nulo o distinto a cualquier identificador de opción de texto añadida.

Notas.

- Sólo se podrá lanzar si el interfaz de conversación se halla activo.

```
void Game.DeactiveConversatorInterfaz(void)
```

*Descripción***.**

- Desactivará de inmediato el interfaz de conversación, inhabilitando el uso de los métodos asociados a añadir opciones u obtenerlas.

Notas.

- Sólo se podrá lanzar si el interfaz de conversación se halla activo.

```
void Game.ShowPresentation(string PresentationProfile)
```

*Descripción***.**

- Mostrará la presentación cuyo perfil coincida con el recibido por parámetro en la llamada al método.

Notas.

- Sólo se podrá lanzar desde el interfaz principal o bien desde el inventario del jugador.

```
void Game.BeginCutScene(string PresentationProfile)
```

*Descripción***.**

- Inicio de modo de cut – scene, en el cual aparecerán dos cortinas de cine e imposibilitarán que el usuario pueda dar órdenes al jugador vía interfaz.

Notas.
- Sólo se podrá lanzar si no está activado el estado de cut – scene y no hay combate activo.

```
void Game.EndCutScene(string PresentationProfile)
```
*Descripción***.**
- Finalización del modo de cut – scene y vuelta al interfaz de usuario normal.

Notas.
- Sólo se podrá lanzar si está activado estado de CutScene.

```
void Game.SetScript(number EventType,
                    string ScriptFile)
```
*Descripción***.**
- Establece un script para un determinado evento.

Parámetros.
- `EventType`. Tipo de evento. Consultar la sección de constantes globales para ver los identificadores de los distintos tipos de eventos.
- `ScriptFile`. Nombre, con extensión, del fichero script a asociar. En caso de querer desasociar cualquier script del evento, se deberá de pasar la cadena vacía (“”).

Notas.
- Sólo se podrán asociar scripts válidos.

F.2. El objeto World.

El objeto `World` se encargará de representar el universo de juego, las distintas áreas, el control global de las entidades de juego, etc. A continuación se lista la relación de los métodos que poseerá y su explicación.

```
string World.GetAreaName(void)
```
*Descripción***.**
- Obtiene el nombre del área en el que el jugador se halla actualmente.

Devolución.
- El nombre del área en el que el jugador se halla.

```
number World.GetAreaID(void)
```
*Descripción***.**
- Obtiene el identificador del área actual.

Devolución.
- El identificador del área actual.

```
number World.GetAreaWidth(void)
```
*Descripción***.**
- Obtiene el número de celdas a lo ancho que tiene el área en el que se halla el jugador actualmente.

Devolución.
- El número de celdas a lo ancho del área actual en el que se halla el jugador.

```
number World.GetAreaHeight(void)
```
*Descripción***.**
- Obtiene el número de celdas a lo alto que tiene el área en el que se halla el jugador actualmente.

Devolución.
- El número de celdas a lo alto del área actual en el que se halla el jugador.

```
number World.GetHour(void)
```
*Descripción***.**
- Obtiene la hora actual en el universo de juego.

Devolución.
- La hora actual del universo de juego.

```
number World.GetMinute(void)
```
*Descripción***.**
- Obtiene el minuto actual en el universo de juego.

Devolución.
- El minuto actual del universo de juego.

```
void World.SetHour(number NewHour)
```
*Descripción***.**
- Establece una nueva hora en el universo de juego.

Parámetros.
- `NewHour`. La nueva hora a establecer.

```
void World.SetMinute(number NewMinute)
```
*Descripción***.**
- Establece un nuevo minuto dentro de la hora en la que se halle el universo de juego.

Parámetros.
- `NewMinute`. Minuto a establecer.

```
entity World.GetEntity(string EntityTag)
```
*Descripción***.**
- Obtiene el handle a la entidad a partir del tag recibido por parámetro. Dicho tag deberá de existir registrado o de lo contrario se retornará la entidad nula.

Parámetros.
- `EntityTag`. Tag asociado a la entidad para la cual se desea recibir el handle.

Devolución.
- Si se pudo localizar el tag, el handle a la entidad a la que pertenezca, en caso contrario la entidad nula (valor `NULL`).

```
entity World.GetPlayer(void)
```

*Descripción***.**
- Obtiene la entidad que apunta al jugador.

Devolución.
- La entidad que representa al jugador.

```
number World.IsFloorValid(number XPos,
                          number YPos)
```

*Descripción***.**
- Comprueba si la posición del mapa recibida es válida o no.

Parámetros.
- `XPos`. Posición en X del área.
- `YPos`. Posición en Y del área.

Devolución.
- Si la posición es válida 1, en caso contrario 0.

```
void World.SetElevationAt(number XPos,
                          number YPos,
                          number Elevation)
```

*Descripción***.**
- Establece la elevacion en la posición dada.

Parámetros.
- `XPos`. Posición en X del área.
- `YPos`. Posición en Y del área.
- `Elevation`. Elevación a establecer.

Notas.
- La posición recibida deberá de corresponder a una celda válida y con contenido.

```
number World.GetElevationAt(number XPos,
                            number YPos)
```

*Descripción***.**
- Obtiene la elevación asociada a la posición del mapa recibida.

Parámetros.
- `XPos`. Posición en X del área.
- `YPos`. Posición en Y del área.

Devolución.
- El valor de la elevación asociada a la posición recibida.

Notas.
- La posición recibida deberá de corresponder a una celda válida y con contenido.

```
number World.GetNumberOfItemsAt(number XPos,
                                number YPos)
```

*Descripción***.**
- El número de ítems que se hallen abandonados sobre una celda del área actual.

Parámetros.
- `XPos`. Posición en X del área.
- `YPos`. Posición en Y del área.

Devolución.
- El número de ítems sobre la celda del área actual recibida.

Notas.
- La posición recibida deberá de corresponder a una celda válida y con contenido.

```
entity World.GetItemAt(number XPos,
                       number YPos,
```

```
                              number EntityPosInFloor)
```

*Descripción***.**

- Obtiene el handle al ítem que se halle en la celda recibida y, dentro de dicha celda, en la posición también pasada.

Parámetros.

- `XPos`. Posición en X del área.
- `YPos`. Posición en Y del área.
- `EntityPosInFloor`. Posición en la que se hallará el ítem de entre todos los posibles ítems que pudieran hallarse sobre la misma celda del área recibida.

Devolución.

- El ítem que cumpla con los requisitos recibidos por parámetro. Si no se halla ninguno, el valor de entidad nula (`NULL`).

Notas.

- La posición recibida deberá de corresponder a una celda válida y con contenido. La posición también deberá de ser válida.

```
number World.GetDistance(number XPosSrc,
                         number YPosSrc,
                         number XPosDest,
                         number XPosSrc)
```

*Descripción***.**

- Obtiene la distancia entre dos celdas del mapa.

Parámetros.

- `XPosSrc`. Posición en X origen del área.
- `YPosSrc`. Posición en Y origen del área.
- `XPosDest`. Posición en X destino del área.
- `YPosDest`. Posición en Y destino del área.

Devolución.

- La distancia entre las dos posiciones recibidas. Si alguna de las dos posiciones recibidas no es válida, se retornará el valor –1.

Notas.

- Las posiciones recibidas deberán de corresponder a celdas válidas y con contenido.

```
number World.CalculePathLenght(number XPosSrc,
                               number YPosSrc,
                               number XPosDest,
                               number XPosSrc)
```

*Descripción***.**

- Calcula la longitud del camino existente entre dos celdas del mapa, siempre y cuando exista dicho camino.

Parámetros.

- `XPosSrc`. Posición en X origen del área.
- `YPosSrc`. Posición en Y origen del área.
- `XPosDest`. Posición en X destino del área.
- `YPosDest`. Posición en Y destino del área.

Devolución.

- Calculará la longitud del camino entre dos posiciones. Si alguna de las dos posiciones no es válida se retornará el valor –1.

Notas.

- Las posiciones recibidas deberán de corresponder a celdas válidas y con contenido.

```
void World.LoadArea(number IDArea,
                    number XPlayerPos,
                    number YPlayerPos)
```

*Descripción***.**

- Cargará un área distinto al actual.

Parámetros.

- `IDArea`. Identificador del área a cargar.
- `XPlayerPos`. Posición en X donde se situará al jugador.

- `YPlayerPos`. Posición en Y donde se situará al jugador.

Notas.

- El identificador del área deberá de ser un identificador válido.

```
number World.ChangeEntityLocation(entity TheEntity,
                                   number XPos,
                                   number YPos)
```

*Descripción***.**

- Cambiará de posición a una entidad.

Parámetros.

- `TheEntity`. Entidad a cambiar de posición.
- `XPos`. Posición en X donde situar a la entidad en el área actual.
- `YPos`. Posición en Y donde situar a la entidad en el área actual.

Devolución.

- Si se pudo efectuar el cambio de posición 1, en caso contrario 0.

Notas.

- La entidad a cambiar deberá de ser válida.

```
void World.AttachCameraToEntity(entity TheEntity,
                                number DoTravelling)
```

*Descripción***.**

- Asociará la cámara a una entidad que podrá ser distinta al jugador. Asociar la cámara significará centrar la vista en la entidad recibida.

Parámetros.

- `TheEntity`. Entidad a la que asociar la cámara.
- `DoTravelling`. Si su valor es igual o inferior a cero no se realizará el efecto de Travelling, asociando la cámara directamente. Sin embargo, si su valor oscila entre 1 y 255 se realizará un scroll hasta la posición donde se halle la entidad.

Notas.

- Se deberá de poseer el modo de cut – scene activado.
- Para evitar problemas relativos de asociar velocidades muy altas o muy bajas, el valor de DoTravelling se acotará automáticamente, siendo la cota inferior 32.

```
void World.AttachCameraToLocation(number XPos,
                                  number YPos,
                                  number DoTravelling)
```

*Descripción***.**

- Asociará la cámara a una posición del área actual. Asociar la cámara significará centrar la vista en la posición recibida.

Parámetros.

- `XPos`. Posición en X del área donde centrar la cámara.
- `YPos`. Posición en Y del área donde centrar la cámara.
- `DoTravelling`. Si su valor es igual o inferior a cero no se realizará el efecto de Travelling, asociando la cámara directamente. Sin embargo, si su valor oscila entre 1 y 255 se realizará un scroll hasta la posición donde se halle la posición.

Notas.

- Se deberá de poseer el modo de cut – scene activado.
- Para evitar problemas relativos de asociar velocidades muy altas o muy bajas, el valor de DoTravelling se acotará automáticamente, siendo la cota inferior 32.

```
number World.IsCombatModeActive(void)
```

*Descripción***.**

- Comprobará si el modo combate se halla, o no, activo.

Devolución.

- Si el modo combate se halla activo 1, en caso contrario 0.

```
void World.EndCombat(void)
```

*Descripción***.**

- Finalizará cualquier tipo de combate que pudiera hallarse activo, liberando el conjunto de entidades definidas como enemigas del jugador.

```
entity World.GetCriatureInCombatTurn(void)
```

*Descripción***.**

- Obtiene la criatura que se halla en el turno de combate actual.

Devolución.

- La entidad con la criatura con el turno de combate actual. Si no se estuviera en modo combate, se retornará el handle nulo (valor `NULL`).

```
entity World.GetCombatant(number Alingment,
                          number CombatantPos)
```

*Descripción***.**

- Obtiene la entidad que se halla combatiendo en la alineación y posición, dentro de la lista de criaturas con su misma alineación, recibidas.

Parámetros.

- `Alingment`. Alineación de combate sobre la que trabajar. Consultar el apartado de constantes globales.
- `CombatantPos`. Posición de la criatura dentro del conjunto de criaturas cuya alineación coincide con la recibida.

Devolución.

- La entidad con la criatura con el turno de combate actual. Si no se estuviera en modo combate, se retornará el handle nulo (valor `NULL`).

Notas.

- La alineación deberá de ser de combate.

```
number World.GetNumberOfCombatants(number Alingment)
```

*Descripción***.**

- Obtiene el número de criaturas que se hallen en la alineación recibida.

Parámetros.

- `Alingment`. Alineación de combate sobre la que consultar el número de combatientes. Consultar el apartado sobre constantes globales.

Devolución.

- El número de combatientes en la alineación recibida.

Notas.

- La alineación deberá de ser de combate.

```
number World.GetAreaLightModel(void)
```

*Descripción***.**

- Obtiene el modelo de luz asociado al área actual.

Devolución.

- El modelo de luz asociado al área actual. Si el valor es de 0, significará que será acorde al paso horario y si es distinto será el valor de luz fijo asociado al área.

```
void World.SetScript(number EventType,
                     string ScriptFile)
```

*Descripción***.**

- Establece un script para un determinado evento.

Parámetros.

- `EventType`. Tipo de evento. Consultar la sección de constantes globales para ver los identificadores de los distintos tipos de eventos.
- `ScriptFile`. Nombre, con extensión, del fichero script a asociar. En caso de querer desasociar cualquier script del evento, se deberá de pasar la cadena vacía (“”).

Notas.

- Se deberá de pasar un script válido para el objeto `World`.

```
void World.SetIdleScriptTime(number Time)
```

*Descripción***.**

- Establece tiempo asociado al evento Idle.

Parámetros.

- `Time`. El tiempo asociado al evento idle del universo de juego.

```
void World.DestroyEntity(entity TheEntity)
```

*Descripción***.**

- Destruye la entidad recibida por parámetro. Al destruirla, desaparecerá completamente del escenario. Cuando se destruya una entidad contenedora de ítems (objetos de escenario contenedores y criaturas), se depositarán sobre la celda que ocupaba los ítems que pudiera contener.

Parámetros.

- `TheEntity`. La entidad que se desea destruir.

Notas.

- La entidad a destruir deberá de ser válida y no ser el jugador.

```
entity World.CreateCriature(number XPos,
                            number YPos,
                            string Profile,
                            string Tag,
                            number Elevation,
                            number Light,
                            number TempCriature)
```

*Descripción***.**

- Creará una criatura sobre el área actual.

Parámetros.

- `XPos`. Posición en X donde situar a la criatura creada sobre el área actual.
- `YPos`. Posición en Y donde situar a la criatura creada sobre el área actual.
- `Profile`. Identificador del perfil de la criatura, para acudir a los archivos de datos y obtener de ellos las características de la misma.
- `Tag`. Posible etiqueta asociada (de no desear asociar ninguna, bastará con pasar la cadena vacía “”).
- `Elevation`. Elevación de la criatura sobre el suelo.
- `Light`. Posible luz asociada.
- `TempCriature`. Si el valor es mayor o igual a 1, la criatura será temporal, lo que querrá decir que cuando se abandone el área actual la criatura será destruida. Si su valor es inferior a 1 la criatura será permanente y continuará existiendo pese a que se abandone el área actual.

Devolución.

- La entidad con la criatura creada. En caso de que no se haya podido crear, la entidad nula (valor `NULL`).

Notas.

- Si no se pudo crear la entidad, se entenderá que algún parámetro no será correcto y el script será interrumpido.

```
entity World.CreateScenaryObject(number XPos,
                                 number YPos,
                                 string Profile,
                                 string Tag,
                                 number Elevation,
                                 number Light)
```

*Descripción***.**

- Creará un objeto de escenario sobre el área actual.

Parámetros.

- `XPos`. Posición en X donde situar al objeto de escenario creado sobre el área actual.
- `YPos`. Posición en Y donde situar al objeto de escenario creado sobre el área actual.
- `Profile`. Identificador del perfil del objeto de escenario, para acudir a los archivos de datos y obtener de ellos las características de la misma.
- `Tag`. Posible etiqueta asociada (de no desear asociar ninguna, bastará con pasar la cadena vacía “”).
- `Elevation`. Elevación del objeto de escenario sobre el suelo.
- `Light`. Posible luz asociada.

Devolución.

- La entidad con el objeto de escenario. En caso de que no se haya podido crear, la entidad

nula (valor `NULL`).

Notas.

- Si no se pudo crear la entidad, se entenderá que algún parámetro no será correcto y el script será interrumpido.

```
entity World.CreateWall(number XPos,
                        number YPos,
                        string Profile,
                        string Tag,
                        number Elevation,
                        number Light,
                        number BlockAccessFlag)
```

*Descripción***.**

- Creará una pared sobre el área actual.

Parámetros.

- `XPos`. Posición en X donde situar a la pared creada sobre el área actual.
- `YPos`. Posición en Y donde situar a la pared creada sobre el área actual.
- `Profile`. Identificador del perfil de la pared, para acudir a los archivos de datos y obtener de ellos las características de la misma.
- `Tag`. Posible etiqueta asociada (de no desear asociar ninguna, bastará con pasar la cadena vacía “”).
- `Elevation`. Elevación de la pared sobre el suelo.
- `Light`. Posible luz asociada.
- `BlockAccessFlag`. Si el valor es mayor o igual a 1, la pared tendrá el acceso bloqueado sobre la celda en la que se sitúo de forma acorde a su orientación. En caso contrario se podrá franquear.

Devolución.

- La entidad con la pared creada. En caso de que no se haya podido crear, la entidad nula (valor `NULL`).

Notas.

- Si no se pudo crear la entidad, se entenderá que algún parámetro no será correcto y el script será interrumpido.

```
entity World.CreateItemAbandoned(number XPos,
                                 number YPos,
                                 string Profile,
                                 string Tag,
                                 number Light)
```

*Descripción***.**

- Creará un ítem sobre una celda del escenario.

Parámetros.

- `XPos`. Posición en X donde situar al ítem creado sobre el área actual.
- `YPos`. Posición en Y donde situar al ítem creado sobre el área actual.
- `Profile`. Identificador del perfil del ítem, para acudir a los archivos de datos y obtener de ellos las características de la misma.
- `Tag`. Posible etiqueta asociada (de no desear asociar ninguna, bastará con pasar la cadena vacía “”).
- `Light`. Posible luz asociada.

Devolución.

- La entidad al ítem. En caso de que no se haya podido crear, la entidad nula (valor `NULL`).

Notas.

- Si no se pudo crear la entidad, se entenderá que algún parámetro no será correcto y el script será interrumpido.

```
entity World.CreateItemWithOwner(entity Owner,
                                 string Profile,
                                 string Tag,
                                 number Light)
```

*Descripción***.**

- Creará un ítem asociado a una entidad contenedora.

Parámetros.

- `Owner`. Entidad que contendrá al ítem creado y que deberá de ser contenedora.
- `Profile`. Identificador del perfil del ítem, para acudir a los archivos de datos y obtener de ellos las características de la misma.
- `Tag`. Posible etiqueta asociada (de no desear asociar ninguna, bastará con pasar la cadena vacía “”).
- `Light`. Posible luz asociada.

Devolución.

- La entidad al ítem. En caso de que no se haya podido crear, la entidad nula (valor `NULL`).

Notas.

- Si no se pudo crear la entidad, se entenderá que algún parámetro no será correcto y el script será interrumpido.

```
void World.SetWorldTimePause(number SetFlag)
```

*Descripción***.**

- Establecerá o quitará la pausa del transcurso horario en el universo de juego. El transcurso horario simplemente hará referencia al paso de las horas.

Parámetros.

- `SetFlag`. Si tiene un valor igual o superior a 1 lo establecerá. En caso contrario quitará la posible pausa.

```
number World.IsWorldTimeInPause(void)
```

*Descripción***.**

- Comprobará si el transcurso horario del universo de juego se halla pausado.

Devuelve.

- Si el transcurso horario del universo de juego está en pausa 1, en caso contrario 0.

```
void World.NextTurn(entity TheCriature)
```

*Descripción***.**

- Pasará el turno de la criatura recibida si, y solo sí, dicha criatura posee el turno actual.

Parámetros.

- `TheCriature`. La criatura para la que pasar el turno.

```
number World.GetLightAt(number XPos,
                        number YPos)
```

*Descripción***.**

- Obtendrá el valor luminoso incidente en una determinada posición. Un valor nulo indicará que no llega nada de luz a la posición suministrada y un valor de 255 indicará que dicha posición está completamente iluminada. Hay que tener en cuenta, que no se tendrá en consideración la luz ambiente global ni el momento del día, simplemente la luz incidente.

Parámetros.

- `XPos, YPos`. La posición del área de juego a consultar.

Devolución.

- La cantidad de luz incidente en la posición suministrada.

```
void World.PlayWAVSound(string szWAVSound)
```

*Descripción***.**

- Método que servirá para reproducir, de forma lineal, un fichero WAV anónimo, esto es, en el universo de juego.

Parámetros.

- `szWAVSound`. El nombre y la extensión del fichero WAV.

```
void World.SetScriptAt(number ScriptType,
                       number XPos,
                       number YPos,
                       string szScript)
```

*Descripción***.**

- Establece un script sobre una posición del área de juego. Al tratarse de una posición, los únicos eventos disponibles serán los correspondientes a `OnSetInFloor` y

`OnSetOutOfFloor`.

Parámetros.

- `ScriptType`. El tipo de script a establecer.
- `XPos, YPos`. La posición donde establecer el script.
- `szScript`. El nombre del script. En caso de pasar la cadena vacía, se quitará cualquier script que estuviera vinculado.

F.3. Métodos comunes a todas las entidades.

A continuación se mostrarán todos aquellos métodos comunes a cualquier tipo de entidad. En algunos casos, estos métodos no valdrán para ciertos tipos de entidades pero serán igualmente puestos aquí debido a que podrán ser utilizando por 2 o más tipos de entidades diferentes.

```
number TheEntity.GetEntityType(void)
```

*Descripción***.**

- Obtiene el tipo de entidad.

Devolución.

- El tipo de entidad (criatura, jugador, objeto de escenario, pared o ítem). Consultar constantes globales para conocer los códigos para identificar a los diferentes tipos de entidad.

```
string TheEntity.GetName(void)
```

*Descripción***.**

- Obtiene el nombre de la entidad.

Devolución.

- El nombre de la entidad.

```
void TheEntity.SetName(string szNewName)
```

*Descripción***.**

- Establece o cambia el nombre asociado a una entidad.

Parámetros.

- `szNewName`. El nombre a establecer.

```
number TheEntity.GetType(void)
```

*Descripción***.**

- Obtiene el tipo de la entidad.

Devolución.

- El tipo de la entidad.

```
number TheEntity.GetClass(void)
```

*Descripción***.**

- Obtiene la clase de la entidad.

Devolución.

- La clase de la entidad.

```
void TheEntity.Say(string WAVFileName,
                   string Text,
                   number TextTime,
                   number RGBTextColor,
                   number TextJustify,
                   number WaitForActionFlag)
```

*Descripción***.**

- Hace que la entidad diga algo.

Parámetros.

- `WAVFileName`. Nombre del archivo WAV asociado. En caso de pasar la cadena vacía, se entenderá que no hay fichero WAV asociado.
- `Text`. Texto que la criatura deberá de decir. Si se pasó un fichero WAV no se tendrá porqué pasar la cadena vacía.
- `TextTime`. En caso de haber pasado texto asociado, el tiempo que éste deberá de permanecer en pantalla.
- `RGBTextColor`. En caso de haber pasado texto asociado, el color del mismo en formato RGB. Consultar las funciones de *API* para conocer cómo obtener un color en formato RGB.
- `TextJustify`. En caso de haber pasado texto asociado, la justificación del mismo con respecto a la criatura. Consultar el apartado de constantes globales para ver posibles justificaciones en el texto.
- `WaitForActionFlag`. Si vale igual o superior a 1, el script esperará hasta que la criatura

finalice de decir algo. En caso contrario la ejecución del script continuará.

```
void TheEntity.ShutUp(void)
```
*Descripción***.**
- En caso de que la entidad se encuentre diciendo algo, la mandará callar.

```
number TheEntity.IsSaying(void)
```
*Descripción***.**
- Comprueba si la entidad se encuentra diciendo algo o no.

Devolución.
- Si la entidad se encuentra diciendo algo, un valor igual a 1. En caso contrario un valor igual a 0.

```
void TheEntity.AttachGFX(string GFXProfile)
```
*Descripción***.**
- Asociará a la entidad un GFX.

Parámetros.
- `GFXProfile`. Perfil del GFX a asociar.

```
void TheEntity.ReleaseGFX(string GFXProfile)
```
*Descripción***.**
- Liberará un perfil de GFX asociado a la entidad. Naturalmente, la entidad deberá de tener asociado un GFX con el perfil recibido.

Parámetros.
- `GFXProfile`. Perfil del GFX a liberar de la entidad.

```
void TheEntity.ReleaseAllGFX(void)
```
*Descripción***.**
- Liberará todos los GFX que pudiera tener asociados una entidad.

```
number TheEntity.IsGFXAttached(string GFXProfile)
```
*Descripción***.**
- Comprueba si el perfil de GFX recibido se halla o no, asociado a la entidad.

Parámetros.
- `GFXProfile`. Perfil del GFX a comprobar si se halla o no asociado.

Devolución.
- Si el perfil se halla asociado 1, en caso contrario 0.

```
number TheEntity.GetNumberOfItemsIn(void)
```
*Descripción***.**
- Obtendrá el número de ítems que poseerá la entidad.

Devolución.
- El número de ítems contenidos en la entidad.

```
entity TheEntity.GetItemAt(number PosInContainer)
```
*Descripción***.**
- Obtendrá el ítem que, hallándose en el interior de la entidad, ocupe la posición recibida.

Devolución.
- La entidad al ítem que ocupa la posición recibida en el interior de la entidad. Si no se hallara, el valor de entidad nula (`NULL`).

```
number TheEntity.IsItemIn(entity TheItem)
```
*Descripción***.**
- Comprueba si un ítem se halla en en un contenedor.

Parámetros.
- `TheItem`. Handle al ítem para el que comprobar si se halla o no en la entidad.

Devolución.
- Si el ítem recibido se halla en el contenedor 1. En caso contrario valor 0.

```
void TheEntity.TransferItemToEntity(entity TheItem,
```

```
                                        entity TheEntityDest)
```

*Descripción***.**

- Pasa un ítem contenido en `TheEntity` a otra entidad diferente.

Parámetros.

- `TheItem`. El ítem que se desea transferir y que debe de ser poseído por `TheEntity`.
- `TheEntityDest`. La entidad destino a la que se desea pasar el ítem.

```
void TheEntity.InsertItemIn(entity TheItem)
```

*Descripción***.**

- Inserta un ítem que se halle sobre una celda del área de juego en la entidad.

Parámetros.

- `TheItem`. El ítem que se desea insertar en la entidad.

Notas.

- La entidad deberá de ser contenedora.

```
void TheEntity.ReleaseItem(entity TheItem)
```

*Descripción***.**

- Hace que una entidad se desprenda del ítem rebido por parámetro. El ítem será despositado en la misma celda en la que se halla la entidad.

Parámetros.

- `TheItem`. El ítem del que la entidad debe de desprenderse.

Notas.

- La entidad deberá de ser contenedora.

```
void TheEntity.SetAnimTemplateState(number State)
```

*Descripción***.**

- Establece un estado de animación sobre la entidad. En caso de que el estado no exista, se ignorará la llamada.

Parámetros.

- `State`. El estado de animación a establecer.

```
void TheEntity.SetPortraitAnimTemplateState(number State)
```

*Descripción***.**

- Establece un estado de animación sobre el posible retrato que tenga asociado la entidad. En caso de que el estado o el retrato no exista, se ignorará la llamada.

Parámetros.

- `State`. El estado de animación a establecer.

```
void TheEntity.SetLight(number Light)
```

*Descripción***.**

- Establece un valor luminoso asociado a la entidad.

Parámetros.

- `Light`. El valor luminoso a establecer. Un valor de 0, hará que la entidad no tenga valor asociado y un valor de 255 significará que tenga el máximo valor luminoso posible.

```
number TheEntity.GetLight(void)
```

*Descripción***.**

- Obtiene el valor luminoso asociado a la entidad.

Devolución.

- El valor luminoso asociado a la entidad.

```
number TheEntity.GetXPos(void)
```

*Descripción***.**

- Obtiene la posición en X de la entidad sobre el área de juego.

Devolución.

- La posición en X de la celda que ocupe la entidad.

```
number TheEntity.GetYPos(void)
```

*Descripción***.**

- Obtiene la posición en Y de la entidad sobre el área de juego.

Devolución.
- La posición en Y de la celda que ocupe la entidad.

```
void TheEntity.SetScript(number EventType,
                         string ScriptFile)
```
*Descripción***.**
- Establece un script para un determinado evento.

Parámetros.
- `EventType`. Tipo de evento. Consultar la sección de constantes globales para ver los identificadores de los distintos tipos de eventos.
- `ScriptFile`. Nombre, con extensión, del fichero script a asociar. En caso de querer desasociar cualquier script del evento, se deberá de pasar la cadena vacía (“”).

```
void TheEntity.SetIdleScriptTime(number Time)
```
*Descripción***.**
- Establece el tiempo asociado al evento Idle.

Parámetros.
- `Time`. El tiempo asociado al evento Idle de la entidad. Caso de pasar 0, se entenderá que se deseará desactivar.

```
number TheEntity.GetElevation(void)
```
*Descripción***.**
- Obtiene la elevación de una entidad.

Devolución.
- La elevación de la entidad (una elevación de cero, significará que la entidad se halla sobre el suelo).

Notas.
- Los ítems no podrán variar su elevación, por lo que siempre devolverá cero.

```
void TheEntity.SetElevation(number Elevation)
```
*Descripción***.**
- Establece la elevación de una entidad.

Parámetros.
- `Elevation`. Elevación a establecer. Un valor de cero significará que la entidad estará sobre el suelo.

Notas.
- Los ítems no podrán variar su elevación, que siempre será cero.

```
number TheEntity.GetLocalAttribute(number IDAttribute)
```
*Descripción***.**
- Obtiene el valor de un atributo local.

Parámetros.
- `IDAttribute`. Identificador del atributo local.

Devolución.
- El valor del atributo local introducido.

Notas.
- Este método no estará disponible para las entidades de tipo criatura pero sí para los ítems, paredes y objetos de escenario.

```
void TheEntity.SetLocalAttribute(number IDAttribute,
                                 number Value)
```
*Descripción***.**
- Establece el valor de un atributo local.

Parámetros.
- `IDAttribute`. Identificador del atributo local.
- `Value`. Valor a establecer.

Notas.
- Este método no estará disponible para las entidades de tipo criatura pero sí para los ítems, paredes y objetos de escenario.

F.4. Métodos particulares de las entidades de tipo ítem.

A continuación se mostrarán los métodos que sólo estarán disponibles para las entidades que sean ítems (ahora `TheEntity` representará a una entidad de tipo ítem).

```
entity TheEntity.GetOwner(void)
```
*Descripción***.**
- Obtiene la entidad que es dueña del ítem.

Devolución.
- La entidad que sea dueña del ítem. En caso de que el ítem no tenga dueño y se halle sobre el área, se retornará la entidad nula (valor `NULL`).

```
number TheEntity.GetGlobalAttribute(number IDAttribute)
```
*Descripción***.**
- Obtiene el valor de un atributo global.

Parámetros.
- `IDAttribute`. Identificador del atributo global.

Devolución.
- El valor del atributo global introducido.

```
void TheEntity.SetGlobalAttribute(number IDAttribute,
                                  number Value)
```
*Descripción***.**
- Establece el valor de un atributo global.

Parámetros.
- `IDAttribute`. Identificador del atributo global.
- `Value`. Valor a establecer.

```
number TheEntity.GetGlobalAttribute(number IDAttribute)
```
*Descripción***.**
- Obtiene el valor de un atributo global.

Parámetros.
- `IDAttribute`. Identificador del atributo global.

Devolución.
- El valor del atributo global introducido.

```
number TheEntity.GetInCombatUseCost(void)
```
*Descripción***.**
- Obtiene el valor de coste en combate. Si el valor fuera cero, significará que el ítem no es un arma y que por lo tanto no puede ser equipada en los slots principales.

Devolución.
- El coste del uso en combate.

F.5. Métodos particulares de las entidades de tipo pared.

A continuación se mostrarán los métodos que sólo estarán disponibles para las entidades que sean paredes (ahora `TheEntity` representará a una entidad de tipo pared).

```
number TheEntity.GetWallOrientation(void)
```
*Descripción***.**
- Obtiene la orientación asociada a la pared.

Devolución.
- La orientación asociada a la pared. Consultar la tabla de constantes globales para conocer los valores numéricos de las distintas orientaciones.

```
void TheEntity.BlockAccess(void)
```
*Descripción***.**
- Bloquea el franqueo a través de la pared según su tipo de orientación.

```
void TheEntity.UnblockAccess(void)
```
*Descripción***.**
- Desbloquea el franqueo a través de la pared según su tipo de orientación.

```
number TheEntity.IsAccessBlocked(void)
```
*Descripción***.**
- Comprueba si la pared se halla bloqueando el acceso.

Devolución.
- Si la pared bloquea el acceso 1, en caso contrario 0.

F.6. Métodos particulares de las entidades de tipo criatura.

A continuación se mostrarán los métodos que sólo estarán disponibles para las entidades que sean criaturas (ahora `TheEntity` representará a una entidad de tipo criatura).

```
void TheEntity.SetSymptom(number IDSymptom,
                          number SetFlag)
```

*Descripción***.**
- Establece o quita un síntoma de la entidad.

Parámetros.
- `IDSymptom`. Identificador del síntoma con el que trabajar. Deberá de ser un identificador numérico acorde a los definidos en el archivo de reglas.
- `SetFlag`. Si su valor es igual o superior a 1, se establecerá y en caso contrario se quitará.

Devolución.
- Si la pared bloquea el acceso 1, en caso contrario 0.

```
number TheEntity.IsSymptomActive(number IDSymptom)
```

*Descripción***.**
- Comprobará si el síntoma recibido por parámetro se halla activo en la entidad o no.

Parámetros.
- `IDSymptom`. Identificador del síntoma con el que trabajar. Deberá de ser un identificador numérico acorde a los definidos en el archivo de reglas.

Devolución.
- Si el síntoma se halla activo en la entidad 1, en caso contrario 0.

```
number TheEntity.GetGenre(void)
```

*Descripción***.**
- Obtendrá el identificador del género asociado a la criatura.

Devolución.
- El identificador del género asociado a la criatura.

```
number TheEntity.GetHealth(number ValueType)
```

*Descripción***.**
- Obtendrá el valor asociado de salud, según sea el tipo de valor del que se desea obtener la información.

Parámetros.
- `ValueType`. El tipo de valor del que se deseará obtener información sobre la salud. Consultar la tabla de constantes globales para conocer los identificadores que indican valor temporal o valor base.

Devolución.
- El valor de salud asociado a la criatura.

```
void TheEntity.SetHealth(number ValueType,
                         number Value)
```

*Descripción***.**
- Establecerá un nuevo valor de salud asociada a uno de los dos tipos posibles de valores (el temporal o el base).

Parámetros.
- `ValueType`. El tipo de valor del que se deseará obtener información sobre la salud. Consultar la tabla de constantes globales para conocer los identificadores que indican valor temporal o valor base.
- `Value`. El valor de salud a establecer.

```
number TheEntity.GetExtendedAttribute(number IDExtendedAttribute,
                                      number ValueType)
```

*Descripción***.**
- Obtendrá el valor asociado a un atributo extendido.

Parámetros.
- `IDExtendedAttribute`. Identificador del atributo extendido. Deberá de ser un identificador numérico acorde a los definidos en el archivo de reglas.

- `ValueType`. El tipo de valor del que se deseará obtener información sobre la salud. Consultar la tabla de constantes globales para conocer los identificadores que indican valor temporal o valor base.

Devolución.

- El valor del atributo extendido.

```
void TheEntity.SetExtendedAttribute(number IDExtendedAttribute,
                                    number ValueType,
                                    number Value)
```

*Descripción***.**

- Establecerá un valor asociado a un atributo extendido, en alguno de sus dos posibles valores (el temporal y el base).

Parámetros.

- `IDExtendedAttribute`. Identificador del atributo extendido. Deberá de ser un identificador numérico acorde a los definidos en el archivo de reglas.
- `ValueType`. El tipo de valor del que se deseará obtener información sobre la salud. Consultar la tabla de constantes globales para conocer los identificadores que indican valor temporal o valor base.
- `Value`. Valor a establecer.

Devolución.

- El valor del atributo extendido.

```
number TheEntity.GetLevel(void)
```

*Descripción***.**

- Obtendrá el nivel asociado a la criatura

Devolución.

- El nivel asociado a la criatura.

```
void TheEntity.SetLevel(number Level)
```

*Descripción***.**

- Establecerá el valor asociado al nivel de la criatura.

Parámetros.

- `Level`. El nivel a establecer en la criatura.

```
number TheEntity.GetExperience(void)
```

*Descripción***.**

- Obtendrá el valor de la experiencia asociada a la criatura.

Devolución.

- La experiencia asociada a la criatura.

```
void TheEntity.SetExperience(number Experience)
```

*Descripción***.**

- Establecerá un valor de experiencia para la criatura.

Parámetros.

- `Experience`. El valor de experiencia a establecer.

Devolución.

- La experiencia asociada a la criatura.

```
number TheEntity.GetInCombatActionPoints(void)
```

*Descripción***.**

- Obtendrá los puntos de acción que tendrá la criatura durante su turno de combate. En caso de que no se esté en combate o la criatura no posea el turno, se retornará un valor nulo.

Devolución.

- Los puntos de acción de la criatura si se halla en su turno de combate.

```
number TheEntity.GetActionPoints(void)
```

*Descripción***.**

- Obtendrá los puntos de acción que tendrá la criatura.

Devolución.

- Los puntos de acción de la criatura.

```
void TheEntity.SetActionPoints(number ActionPoints)
```
*Descripción***.**
- Establecerá los puntos de combate de la criatura.

Parámetros.
- `ActionPoints`. Los puntos de combate a establecer.

```
number TheEntity.IsHabilityActive(number IDHability)
```
*Descripción***.**
- Comprobará si el identificador de la habilidad recibida se halla activa en la criatura.

Parámetros.
- `IDHability`. El identificador de la habilidad a comprobar. Deberá de oscilar entre los identificadores definidos en el archivo de reglas.

```
void TheEntity.SetHability(number IDHability,
                           number SetFlag)
```
*Descripción***.**
- Establecerá o quitará el identificador de habilidad recibido de la criatura.

Parámetros.
- `IDHability`. El identificador de la habilidad a comprobar. Deberá de oscilar entre los identificadores definidos en el archivo de reglas.
- `SetFlag`. Si su valor es igual o superior a 1, la habilidad se establecerá. En caso contrario se quitará.

```
void TheEntity.UseHability(number IDHability,
                           entity TheEntityTarget,
                           number WaitForActionFlag)
```
*Descripción***.**
- Ordenará a la criatura que utilice la habilidad indicada por parámetro sobre una entidad destino.

Parámetros.
- `IDHability`. El identificador de la habilidad a comprobar. Deberá de oscilar entre los identificadores definidos en el archivo de reglas.
- `TheEntityTarget`. La entidad sobre la que utilizar la habilidad.
- `WaitForActionFlag`. Si su valor es igual o superior a 1, el script se detendrá hasta que no se haya utilizado la habilidad. En caso contrario, continuará con su ejecución.

```
number TheEntity.IsRunModeActive(void)
```
*Descripción***.**
- Comprobará si el modo correr se halla activo en la criatura.

Devolución.
- Si el modo correr se halla activo un valor de 1, en caso contrario 0.

```
void TheEntity.SetRunMode(number SetFlag)
```
*Descripción***.**
- Establece o quita el modo correr en la criatura.

Parámetros.
- `SetFlag`. Si su valor es igual o superior a 1, se establecerá el modo correr. En caso contrario se quitará.

```
number TheEntity.MoveTo(number XPos,
                        number YPos,
                        number WaitForActionFlag)
```
*Descripción***.**
- Ordena a la criatura moverse a una posición del área.

Parámetros.
- `XPos`. Posición en X del área donde mover a la criatura.
- `YPos`. Posición en Y del área donde mover a la criatura.
- `WaitForActionFlag`. Si su valor es igual o superior a 1 se pausará el script hasta que la acción de mover finalice. En caso contrario no.

Devolución.
- Si se ha podido mover a la criatura 1, en caso contrario 0.

Notas.
- Se deberá estar en interfaz principal.

```
number TheEntity.IsMoving(void)
```
Descripción.
- Comprueba si la criatura se encuentra en movimiento.

Devolución.
- Si la criatura se halla en movimiento 1, en caso contrario 0.

```
void TheEntity.StopMoving(void)
```
Descripción.
- Ordena a la criatura que se detenga si es que está moviéndose. Si el movimiento era fruto de una orden para realizar algún tipo de acción, la orden de realización de esa acción finalizará también.

```
void TheEntity.EquipItem(number IDSlot,
                         entity TheItem)
```
Descripción.
- Ordena a la criatura equiparse un ítem en un determinado slot. El ítem ha de poseerlo.

Parámetros.
- `IDSlot`. Identificador del slot donde equipar el ítem. El valor deberá de oscilar entre los identificadores definidos en el archivo de reglas.
- `TheItem`. El ítem que se desea equipar.

```
void TheEntity.RemoveItemEquipped(number IDSlot)
```
Descripción.
- Ordena a la criatura quitarse un ítem previamente equipado un determinado slot.

Parámetros.
- `IDSlot`. Identificador del slot donde equipar el ítem. El valor deberá de oscilar entre los identificadores definidos en el archivo de reglas.

```
entity TheEntity.GetItemEquipped(number IDSlot)
```
Descripción.
- Obtiene el ítem que la criatura posee equipada en un determinado slot.

Parámetros.
- `IDSlot`. Identificador del slot donde se quiere obtener el ítem equipado. El valor deberá de oscilar entre los identificadores definidos en el archivo de reglas.

Devolución.
- El ítem equipado en el slot pasado o la entidad nula (valor NULL) si no hubiera ítem alguno equipado en dicho slot.

```
number TheEntity.IsItemEquipped(number IDSlot)
```
Descripción.
- Comprueba si una criatura posee un ítem equipado en un determinado slot.

Parámetros.
- `IDSlot`. Identificador del slot donde se quiere realizar la comprobación. El valor deberá de oscilar entre los identificadores definidos en el archivo de reglas.

Devolución.
- Si la criatura posee un ítem equipado en el slot pasado un valor de 1, en caso contrario un valor de 0.

```
void TheEntity.DropItem(entity TheItem)
```
Descripción.
- Ordena a la criatura que suelte un ítem. El ítem, además de poseerse, no ha de hallarse equipado. Al soltarlo, el ítem aparecerá en la misma celda en donde se halle la criatura.

Parámetros.
- `TheItem`. El ítem a soltar.

```
void TheEntity.UseItem(entity TheItem,
                       entity TheEntityTarget,
                       number WaitForActionFlag)
```

*Descripción***.**

- Ordena a una criatura utilizar un ítem sobre una entidad destino. El ítem, además de poseerlo, no deberá de hallarse equipado.

Parámetros.

- `TheItem`. El ítem a utilizar.
- `TheEntityTarget`. La entidad sobre la que utilizar el ítem.
- `WaitForActionFlag`. Si su valor es igual o superior a 1 se pausará el script hasta que la acción de usar ítem finalice. En caso contrario no.

Notas.

- Se deberá estar en interfaz principal si se desea usar el ítem sobre otra entidad que no sea la propia.

```
void TheEntity.Manipulate(entity TheEntityTarget,
                          number WaitForActionFlag)
```

*Descripción***.**

- Ordenará a una criatura manipular una entidad.

Parámetros.

- `TheEntityTarget`. La entidad que se desea manipular.
- `WaitForActionFlag`. Si su valor es igual o superior a 1 se pausará el script hasta que la acción de manipular finalice. En caso contrario no.

Notas.

- Se deberá estar en interfaz principal.

```
void TheEntity.SetTransparentMode(number SetFlag)
```

*Descripción***.**

- Establecerá o quitará el modo de transparencia en una criatura.

Parámetros.

- `SetFlag`. Si su valor es igual o superior a 1 se establecerá, en caso contrario se quitará.

```
bool TheEntity.IsTransparentModeActive(void)
```

*Descripción***.**

- Comprobará si el modo de transparencia se halla activo en la criatura.

Devolución.

- Si el modo transparencia se halla activo 1, en caso contrario 0.

```
void TheEntity.ChangeAnimOrientation(number Orientation)
```

*Descripción***.**

- Cambiará la orientación de la criatura a la especificada por parámetro.

Parámetros.

- `Orientation`. La orientación a la que cambiar la criatura. Consultar la tabla de constantes globales para obtener una relación de los valores numéricos asociados a las orientaciones.

```
number TheEntity.GetAnimOrientation(void)
```

*Descripción***.**

- Obtendrá la orientación asociada a la criatura

Devolución.

- La orientación asociada a la criatura. Consultar la tabla de constantes globales para obtener una relación de los valores numéricos asociados a las orientaciones.

```
void TheEntity.SetAlingment(number Alingment)
```

*Descripción***.**

- Establecerá la alineación recibida por parámetro en la criatura.

Parámetros.

- `Alingment`. La alineación a establecer en la criatura. Consultar la tabla de constantes globales para obtener una relación de los valores numéricos asociados a las alineaciones.

Notas.

- Una criatura solo se podrá alinear en combate si alguna de las criaturas ya alineadas,

independientemente de su alineación, se hallan en el rango de la misma.

```
void TheEntity.SetAlingmentWith(entity TheFriendCriature)
```
*Descripción***.**
- Hará que la criatura se alinee con la recibida por parámetro.

Parámetros.
- `TheFriendCriature`. La criatura con la que alinear `TheEntity`.

Notas.
- Una criatura solo se podrá alinear en combate si alguna de las criaturas ya alineadas, independientemente de su alineación, se hallan en el rango de la misma.

```
void TheEntity.SetAlingmentAgainst(entity TheEnemyCriature)
```
*Descripción***.**
- Hará que la criatura se alinee contra la recibida por parámetro.

Parámetros.
- `TheEnemyCriature`. La criatura contra la que alinear `TheEntity`.

Notas.
- Una criatura solo se podrá alinear en combate si alguna de las criaturas ya alineadas, independientemente de su alineación, se hallan en el rango de la misma.

```
number TheEntity.GetAlingment(void)
```
*Descripción***.**
- Obtiene la alineación de la criatura.

Devolución.
- La alineación de la criatura. Consultar la tabla de constantes globales para obtener una relación de los identificadores numéricos.

```
void TheEntity.HitEntity(entity TheEntityTarget,
                         number WeaponSlot,
                         number WaitForActionFlag)
```
*Descripción***.**
- Ordena a una criatura utilizar un ítem, o no, equipado en uno de sus slots principales para golpear a otra entidad.

Parámetros.
- `TheEntityTarget`. La entidad a la que golpear.
- `WeaponSlot`. Identificador del slot principal a utilizar para golpear.
- `WaitForActionFlag`. Si su valor es igual o superior a 1 se pausará el script hasta que la acción de golpear finalice. En caso contrario no.

Notas.
- Se deberá estar en interfaz principal.

```
number TheEntity.IsGhostMoveModeActive(void)
```
*Descripción***.**
- Comprueba si la criatura tiene activo el modo fantasma de movimiento. El modo fantasma de movimiento, permite que la criatura pueda moverse independientemente de las limitaciones de acceso de las celdas.

Devolución.
- Si tiene activo el modo fantasma 1, en caso contrario 0.

```
void TheEntity.SetGhostMoveMode(number SetFlag)
```
*Descripción***.**
- Establece o quita el modo fantasma de movimiento en la criatura. El modo fantasma de movimiento, permite que la criatura pueda moverse independientemente de las limitaciones de acceso de las celdas.

Parámetros.
- `SetFlag`. Un valor mayor o igual a 1 lo establecerá y un valor inferior lo quitará.

```
number TheEntity.GetRange(void)
```
*Descripción***.**
- Obtiene el rango de la criatura.

Devolución.
- El rango de la criatura.

Notas.
- El rango es un valor que se estableció junto al perfil mediante el que fue creado la criatura.

```
number TheEntity.IsInRange(entity TheCriature)
```

*Descripción***.**
- Comprueba si la criatura pasada se halla dentro del rango de la criatura actual.

Parámetros.
- `TheCriature`. El handle a la criatura para la que comprobar si se halla dentro del rango de la criatura actual.

Devolución.
- Si se halla dentro del rango devolverá 1, en caso contrario 0.

F.7. Funciones del API.

```
number APIPassToRGBColor(number RedColor,
                         number GreenColor,
                         number BlueColor)
```

*Descripción***.**

- Realiza la conversión de los componentes RGB de color a un valor numérico usable por aquellos métodos que necesiten recibir por parámetro un valor de color. Los valores RGB de color representarán los componentes rojo, verde y azul de color respectivamente.

Parámetros.

- `RedColor`. Componente rojo de color.
- `GreenColor`. Componente verde de color.
- `BlueColor`. Componente azul de color.

Devolución.

- El valor numérico de los componentes RGB.

```
number APIGetRedComponent(number RGBColor)
```

*Descripción***.**

- Extrae el componente rojo de un valor numérico creado con la función `APIPassToRGBColor`.

Parámetros.

- `RGBColor`. El valor RGB de color.

Devolución.

- El componente rojo embebido en el mismo.

```
number APIGetGreenComponent(number RGBColor)
```

*Descripción***.**

- Extrae el componente verde de un valor numérico creado con la función `APIPassToRGBColor`.

Parámetros.

- `RGBColor`. El valor RGB de color.

Devolución.

- El componente verde embebido en el mismo.

```
number APIGetBlueComponent(number RGBColor)
```

*Descripción***.**

- Extrae el componente azul de un valor numérico creado con la función `APIPassToRGBColor`.

Parámetros.

- `RGBColor`. El valor RGB de color.

Devolución.

- El componente azul embebido en el mismo.

```
number APIRand(number MinValue,
               number MaxValue)
```

*Descripción***.**

- Obtiene un valor aleatorio en el intervalo recibido.

Parámetros.

- `MinValue`. El valor mínimo del intervalo del que obtener un valor aleatorio.
- `MaxValue`. El valor máximo del interfaz del que obtener el valor aleatorio.

Devolución.

- Un valor aleatorio entre `MinValue` y `MaxValue`, incluyéndose ambos.

```
number APIGetIntegerValue(number Value)
```

*Descripción***.**

- Obtiene de un valor numérico su parte entera.

Parámetros.

- `Value`. El valor numérico del que obtener la parte entera.

Devolución.

- La parte entera del valor numérico recibido.

```
number APIGetDecimalValue(number Value)
```
*Descripción***.**
- Obtiene de un valor numérico su parte decimal.

Parámetros.
- `Value`. El valor numérico del que obtener la parte decimal.

Devolución.
- La parte decimal del valor numérico recibido.

```
number APIGetStringSize(string Text)
```
*Descripción***.**
- Obtiene el tamaño del `string` recibido.

Parámetros.
- `Text`. La cadena de caracteres de donde obtener el tamaño.

Devolución.
- El tamaño de la cadena de caracteres.

```
void APIWriteToLogger(string Text)
```
*Descripción***.**
- Escribe datos al archivo "`Logger.txt`".

Parámetros.
- `Text`. El texto a escribir.

```
void APIShowFPS(number ShowFlag)
```
*Descripción***.**
- Muestra o esconde la tasa de frames por segundo (FPS).

Parámetros.
- `ShowFlag`. Si su valor es igual o superior a 1 mostrará la tasa de FPS, en caso contrario los ocultará.

```
void APIWait(number Time)
```
*Descripción***.**
- Hace que el script se pause los segundos recibidos por parámetro. Se podrán indicar valores inferiores a un segundo.

Parámetros.
- `Time`. Los segundos que se desean pausar el script.

```
void APIEnableCrisolScriptWarnings()
```
*Descripción***.**
- Habilitará el envío de mensajes de traza sobre los scripts ejecutados a un archivo llamado "`CrisolScriptWarnings.txt`". Este archivo será de utilidad para el desarrollador de *Crisol* no para el diseñador que quiera realizar un juego.

```
void APIDisableCrisolScriptWarnings()
```
*Descripción***.**
- Deshabilitará el envío de mensajes de traza sobre los scripts ejecutados a un archivo llamado "`CrisolScriptWarnings.txt`". Este archivo será de utilidad para el desarrollador de *Crisol* no para el diseñador que quiera realizar un juego.