**SheRBA API Documentation**

*Version 0.1.0 pre-alpha development*

*2017-08-07*

**General**

* Interfaces
* [IState](#IState)
* Classes
* [GameManager](#GameManager)
* [LocationNode](#LocationNode)
* [Module](#Module)

**Buttons**

* Classes
* [ClickableObject](#ClickableObject)
* [ReturnBtn](#ReturnBtn)
* [SherlockStudyBtn](#SherlockStudyBtn)
* [StartBtn](#StartBtn)

**Inventory**

* Classes
* [AppraiseSlot](#AppraiseSlot)
* [CloseInvBtn](#CloseInvBtn)
* [DraggableObject](#DraggableObject)
* [EvidenceDock](#EvidenceDock)
* [Inventory](#Inventory)
* [InvSlot](#InvSlot)
* [Item](#Item)
* [OpenInvBtn](#OpenInvBtn)
* [SubmitBtn](#SubmitBtn)

**Sherlock**

* Classes
* [Sherlock](#Sherlock)
* [SpeechBubble](#SpeechBubble)

**Util**

* Classes
* [MsgBanner](#MsgBanner)
* [TriggerOnce](#TriggerOnce)

**VideoPlayerConsole**

* Enumerables
* [EVideoState](#EVideoState)
* Classes
* [PlayBtn](#PlayBtn)
* [VideoPlayerConsole](#VideoPlayerConsole)
* [VideoSlider](#VideoSlider)

**IState**

* Summary
* Gives the implementing class a state of being.
* Parent Interface
* Public Methods
* ResetState, void
* Summary
* Resets the implementing object’s variables back to their defaults.

**GameManager**

* Summary
* Master control object for many of the game’s shared and global elements.
* Parent Class
* MonoBehaviour
* Interfaces
* Constants
* Properties
* Fields
* canvas, public GameObject
* fadeFilter, public GameObject
* currentModule, private GameObject
* Summary
* The module that is currently loaded.
* startModule, public GameObject
* Summary
* The module that is loaded at start.
* Remarks
* This Game Object itself still needs to be set as active in the Inspector and all other modules need to be set as inactive.
* Static Members
* inst, [GameManager](#GameManager)
* Summary
* Singleton instance.
* Remarks
* Depreciated in favor of direct referencing in the Inspector.
* msgBanner, GameObject
* Summary
* Singleton instance
* Remarks
* Depreciated in favor of direct referencing in the Inspector.
* Constructor
* Static Methods
* Public Methods
* SetNewModule, void
* Summary
* Unloads the current module and loads the given one.
* Params
* module, GameObject
* RecursiveReset, void
* Summary
* Recursively iterates through every single game object in a module and calls their Reset method in their IState interface.
* Params
* node, GameObject
* Protected Methods
* Private Methods
* Start, void
* Awake, void

**LocationNode**

* Summary
* Event handler for location nodes on the world map.
* Parent Class
* [ClickableObject](#ClickableObject)
* Interfaces
* Constants
* Properties
* Fields
* siteModule, public GameObject
* textPopup, public GameObject
* Static Members
* Constructor
* Static Methods
* Public Methods
* ResetState, override void
* OnPointerClick, override void
* Summary
* Load the module assigned here.
* Params
* ptrData, PointerEventData
* OnPointerEnter, override void
* Summary
* Activate tooltip
* Params
* ptrData, PointerEventData
* OnPointerExit, override void
* Summary
* Deactivate tooltip
* Params
* ptrData, PointerEventData
* Protected Methods
* Private Methods

**Module**

* Summary
* Logic controller for module that toggle Sherlock’s appearance
* Parent Class
* MonoBehaviour
* Interfaces
* Constants
* Properties
* Fields
* sherlock, public GameObject
* Static Members
* Constructor
* Static Methods
* Public Methods
* Protected Methods
* Private Methods
* OnEnable, void
* OnDisable, void

**Buttons**

**ClickableObject**

* Summary
* Base object class that contains functionality for objects that can be clicked. This object is intended to be inherited by another class that provides additional functionality to the below methods.
* Parent Class
* MonoBehaviour
* Interfaces
* [IState](#IState)
* IPointerClickHandler
* IPointerDownHandler
* IPointerUpHandler
* IPointerEnterHandler
* IPointerExitHandler
* Constants
* Properties
* Fields
* hoverColor, public readonly Color32
* Summary
* Default highlight color on mouse over.
* staticColor, protected Color32
* Summary
* Default static color.
* Static Members
* Constructor
* Static Methods
* Public Methods
* ResetState
* Summary
* Implemented methods that resets the variable to their default on this object.
* OnPointerClick, void virtual
* Summary
* No default implementation.
* Params
* ptrData, PointerEventData
* OnPointerEnter, void virtual
* Summary
* Changes the button color.
* Remarks
* The base method should always be called first before the overridden implementation
* Params
* ptrData, PointerEventData
* OnPointerExit, void virtual
* Summary
* Changes the button color back to the default.
* Remarks
* The base method should always be called first before the overridden implementation
* Params
* ptrData, PointerEventData
* OnPointerUp, void virtual
* Summary
* No default implementation
* Params
* ptrData, PointerEventData
* OnPointerDown, void virtual
* Summary
* No default implementation
* Params
* ptrData, PointerEventData
* Protected Methods
* Private Methods
* Start
* Summary
* Cache the default color that is assigned to the image component on this game object.

**ReturnBtn**

* Summary
* Button object that calls [GameManager](#GameManager) to unload the current module, and load the one assigned to this button.
* Remarks
* Shares too many similarities with [ReturnBtn](#ReturnBtn), [StartBtn](#StartBtn), and [SherlockStudyBtn](#SherlockStudyBtn). Needs to be condensed into a single load module class.
* Parent Class
* [ClickableObject](#ClickableObject)
* Interfaces
* Constants
* Properties
* Fields
* manager, public [GameManager](#GameManager)
* module, public GameObject
* Static Members
* Constructor
* Static Methods
* Public Methods
* OnPointerClick, override void
* Summary
* Set the new module to load.
* Params
* ptrData, PointerEventData
* Protected Methods
* Private Methods

**SherlockStudyBtn**

* Summary
* Button object that calls [GameManager](#GameManager) to unload the current module, and load the one assigned to this button.
* Remarks
* Shares too many similarities with [ReturnBtn](#ReturnBtn), [StartBtn](#StartBtn), and [SherlockStudyBtn](#SherlockStudyBtn). Needs to be condensed into a single load module class.
* Parent Class
* [ClickableObject](#ClickableObject)
* Interfaces
* Constants
* Properties
* Fields
* manager, public [GameManager](#GameManager)
* studyModule, public GameObject
* Static Members
* Constructor
* Static Methods
* Public Methods
* OnPointerClick, override void
* Summary
* Set the new module to load.
* Params
* ptrData, PointerEventData
* Protected Methods
* Private Methods

**StartBtn**

* Summary
* Button object that calls [GameManager](#GameManager) to unload the current module, and load the one assigned to this button.
* Remarks
* Shares too many similarities with [ReturnBtn](#ReturnBtn), [StartBtn](#StartBtn), and [SherlockStudyBtn](#SherlockStudyBtn). Needs to be condensed into a single load module class.
* Parent Class
* Interfaces
* Constants
* Properties
* Fields
* manager, public [GameManager](#GameManager)
* worldMap, public GameObject
* Static Members
* Constructor
* Static Methods
* Public Methods
* OnPoniterClick, override void
* Summary
* Set the new module to load.
* Params
* ptrData, PointerEventData
* Protected Methods
* Private Methods

**Inventory**

**AppraiseSlot**

* Summary
* Container for items that Sherlock can appraise.
* Parent Class
* [InvSlot](#InvSlot)
* Interfaces
* Constants
* Properties
* Fields
* sherlock, public [Sherlock](#Sherlock)
* Static Members
* Constructor
* Static Methods
* Public Methods
* OnDrop, override void
* Summary
* Takes the item dropped into this slot and passes them to a method on Sherlock.
* Protected Methods
* Private Methods

**CloseInvBtn**

* Summary
* Button that closes the window object that is assigned here.
* Parent Class
* Interfaces
* Constants
* Properties
* Fields
* window, public GameObject
* Summary
* Object that is toggled off.
* Static Members
* Constructor
* Static Methods
* Public Methods
* OnPointerEnter, override void
* Summary
* Sets the highlight color to red.
* Params
* ptrData, PointerEventData
* Protected Methods
* Private Methods

**DraggableObject**

* Summary
* Base class for any kind of object that can be clicked and dragged.
* Parent Class
* MonoBehaviour
* Interfaces
* IBeginDragHandler
* IDragHandler
* IEndDragHandler
* Constants
* Properties
* Fields
* clickPosition, protected Vector2
* Summary
* Where the cursor was when a drag operation started.
* Static Members
* Constructor
* Static Methods
* Public Methods
* OnBeginDrag, virtual void
* Summary
* Save the position the cursor was at when this is called and block all raycasts on the object being dragged.
* Params
* ptrData, PointerEventData
* OnDrag, virtual void
* Summary
* Updates the dragging object’s potion to the cursor’s position every tick.
* Params
* ptrData, PointerEventData
* OnEndDrag, virtual void
* Summary
* Unblock raycasts when this object is dropped.
* Params
* ptrData, PointerEventData
* Protected Methods
* Private Methods

**EvidenceDock**

* Summary
* Item container that spawns the items at specific time cues for the user’s inventory while the video plays.
* Parent Class
* MonoBehaviour
* Interfaces
* Constants
* Properties
* Fields
* msgBanner, public [MsgBanner](#MsgBanner)
* Summary
* Message Banner object that is called to display alerts.
* canvas, public GameObject
* Summary
* Reference to the main Canvas.
* console, public [VideoPlayerConsole](#VideoPlayerConsole)
* Summary
* Reference to the video player that this object gets current play time values from.
* inventory, public [Inventory](#Inventory)
* Summary
* Main inventory for the player that is called to create new items.
* evidence, public GameObject[]
* Summary
* List of objects that are cloned to the dock and the main inventory.
* Remarks
* Each object shares the same respective index with timeCues and hasSpawned.
* timeCues, public double[]
* Summary
* List of values in seconds that indicate at which time during the video an item from evidence, at the same respective index, will spawn for the dock and the main inventory.
* hasSpawned, private bool[]
* Summary
* List of Booleans that indicate if an object from evidence has been spawned before from the same respective index.
* Remarks
* These items should only ever spawn once per video.
* Static Members
* Constructor
* Static Methods
* Public Methods
* Protected Methods
* Private Methods
* Start, void
* Summary
* Initializes hasSpawned to the length of evidence.
* FixedUpdate, void
* Summary
* Iterates through all items in evidence and attempts to spawn each one if the video time has exceeded their respective time cues and if the respective item has not spawned yet.

**Inventory**

* Summary
* Container of a list of InvSlot that holds items.
* Parent Class
* [DraggableObject](#DraggableObject)
* Interfaces
* Constants
* Properties
* Fields
* isDraggable, public bool
* Summary
* Can the window be dragged and positioned?
* canvas, public GameObject
* Summary
* Reference to the main canvas.
* msgBanner, public [MsgBanner](#MsgBanner)
* Summary
* Message Banner that is called to display alerts.
* slotList, public [InvSlot](#InvSlot)[]
* Summary
* List of inventory slots in this inventory window.
* Remarks
* Each slot needs to be initialized with a reference to a Message Banner.
* Static Members
* Constructor
* Static Methods
* Public Methods
* OnDrag, override void
* Summary
* Allows the window to be draggable.
* Params
* ptrData, PointerEventData
* CreateItem, void
* Summary
* Takes an existing item and clones it into the first available slot in this inventory. Returns an error if inventory is full.
* Params
* item, GameObject
* Protected Methods
* Private Methods
* Start, void
* Summary
* Initializes each inventory slot with a reference to this Message Banner.
* OnEnable, void
* Summary
* Initializes each inventory slot with a reference to this Message Banner if not already.

**InvSlot**

* Summary
* Container for a single item that is displayed in an inventory window.
* Parent Class
* MonoBehaviour
* Interfaces
* IDropHandler
* Constants
* Properties
* Fields
* item, public [Item](#Item)
* Summary
* The item that occupies this slot.
* msgBanner, public [MsgBanner](#MsgBanner)
* Summary
* Message Banner that is called to display alerts.
* Static Members
* Constructor
* Static Methods
* Public Methods
* OnDrop, virtual void
* Summary
* Set the dropped item to this slot if another item doesn’t already occupy.
* Protected Methods
* Private Methods

**Item**

* Summary
* Base item class for objects that can be stored in an inventory windows and can be dragged around.
* Parent Class
* [DraggableObject](#DraggableObject)
* Interfaces
* Constants
* Properties
* Fields
* itemName, public string
* Summary
* Name of this item.
* canvas, public GameObject
* Summary
* Reference to the main Canvas.
* curSlot
* Summary
* The current inventory slot this item is parented to.
* Attributes
* Hide In Inspector
* prevSlot
* Summary
* The previous slot this item was parented to during a drag operation.
* Attributes
* Hide In Inspector
* apprasialSpeech, public string
* Summary
* Path name of the file that holds Sherlock’s appraisal dialogue for this item.
* Static Members
* Constructor
* Static Methods
* Public Methods
* OnBeginDrag, override void
* Summary
* Prepares the item for a drag operation
* Params
* ptrData, PointerEventData
* OnEndDrag
* Summary
* Sets this item to a new slot if the slot is available.
* Params
* ptrData, PointerEventData
* Protected Methods
* Private Methods

**OpenInvBtn**

* Summary
* Button that toggles the referenced window visibility.
* Parent Class
* [ClickableObject](#ClickableObject)
* Interfaces
* Constants
* Properties
* Fields
* window, public GameObject
* Summary
* Object that will be toggled.
* Static Members
* Constructor
* Static Methods
* Public Methods
* OnPointerClick, override void
* Summary
* Toggles window visibility.
* Protected Methods
* Private Methods

**SubmitBtn**

* Summary
* Special kind of button that holds a list to multiple inventories that are submitted to Sherlock for him to analyze.
* Parent Class
* [ClickableObject](#ClickableObject)
* Interfaces
* Constants
* Properties
* Fields
* invList, public [Inventory](#Inventory)[]
* Summary
* List of inventories that are submitted.
* sherlock, public [Sherlock](#Sherlock)
* Summary
* Reference to Sherlock to trigger dialogue.
* Static Members
* Constructor
* Static Methods
* Public Methods
* OnPointerClick, override void
* Summary
* Submits the evidence to Sherlock.
* Protected Methods
* Private Methods

**Sherlock**

**Sherlock**

* Summary
* Handles all of Sherlock’s dialogue and triggerable events.
* Parent Class
* MonoBehaviour
* Interfaces
* Constants
* Properties
* Fields
* speechBubble, public GameObject
* Summary
* Speech bubble object that is toggled on and off.
* bubbleText, public Text
* Summary
* Text field that displays text.
* Static Members
* Constructor
* Static Methods
* Public Methods
* DoSpeech, void
* Summary
* Displays the dialogue from a specific file.
* Params
* fileName, string
* DoReview, void
* Summary
* Displays the dialogue from a string.
* Params
* Text, string
* Clear, void
* Summary
* Clears the text field.
* SubmitEvidence, void
* Summary
* Iterates through the given list of inventories where Sherlock displays a specific dialogue for each item and how it was classified.
* Params
* invList, [Inventory](#Inventory)[]
* Protected Methods
* Private Methods
* DoDialogue, IEnumerator
* Summary
* Coroutine that displays Sherlock’s dialogue one line at a time until the user gives input to continue.
* Remarks
* (WIP)

**SpeechBubble**

* Summary
* Speech bubble systems that controls the visibility and content of the lower third speech bubble form Sherlock.
* Parent Class
* [ClickableObject](#ClickableObject)
* Interfaces
* Constants
* Properties
* Fields
* sherlock, public [Sherlock](#Sherlock)
* Summary
* Reference to Sherlock.
* Static Members
* Constructor
* Static Methods
* Public Methods
* OnPointerClick, void
* Summary
* Resets the speech bubble.
* Params
* ptrData, PointerEventData
* Protected Methods
* Private Methods

**Util**

**MsgBanner**

* Summary
* Displays a timed GUI alert or message
* Parent Class
* MonoBehaviour
* Interfaces
* Constants
* Properties
* Fields
* heading, public Text
* message, public Text
* timer, private [CountdownTimer](NanoTimersDoc_0-1-1d.docx)
* Static Members
* Constructor
* Static Methods
* Public Methods
* SetMessage
* Summary
* Sets the message banner content
* Params
* head, string
* msg, string
* Protected Methods
* Private Methods

**TriggerOnce**

* Summary
* Flow control that allows a block of code to only be triggered once.
* Parent Class
* Interfaces
* Constants
* Properties
* Fields
* triggered, private bool
* Static Members
* Constructor
* Params
* state, bool
* Static Methods
* Public Methods
* Trigger, bool
* Summary
* Returns false if this trigger hasn’t already been triggered.
* Remarks
* Best used in an if-statement.
* Reset
* Summary
* Resets the trigger to allow another execution.
* Protected Methods
* Private Methods

**VideoPlayerConsole**

**EVideoState**

* Summary
* Different states that represent a video player.
* Members
* stopped
* playing
* paused
* seeking

**PlayBtn**

* Summary
* Button that toggles the playing and paused states of a video player.
* Parent Class
* [ClickableObject](#ClickableObject)
* Interfaces
* Constants
* Properties
* Fields
* console, public [VideoPlayerConsole](#VideoPlayerConsole)
* Summary
* Reference to the video player this button controls.
* Static Members
* Constructor
* Static Methods
* Public Methods
* OnPointerClick, override void
* Summary
* Click event that toggles the video player state.
* Params
* ptrData, PointerEventData
* Protected Methods
* Private Methods

**VideoPlayerConsole**

* Summary
* Main video class that controls video playback state and video display properties.
* Parent Class
* MonoBehaviour
* Interfaces
* Constants
* VID\_WIDTH, int
* Value
* 1280
* VID\_HEIGHT, int
* Value
* 720
* VID\_CDEPTH, int
* Value
* 24
* Properties
* State, [EVideoState](#EVideoState)
* Summary
* Changes the state of the video player.
* Remarks
* Resets the [TriggerOnce](#TriggerOnce) in the state machine when changed.
* Fields
* videoPlayer, public VideoPlayer
* videoRenderer, public RawImage
* audioSource, public AudioSource
* pauseFilter, public GameObject
* btnText, public Text
* timeText, public Text
* timeLine, public Slider
* debugTimeText, public Text
* debugFramerText, public Text
* state, private [EVideoState](#EVideoState)
* triggerOnce, private [TriggerOnce](#TriggerOnce)
* Static Members
* Constructor
* Static Methods
* FormatTime, string
* Summary
* Takes in seconds and formats to a string in minutes and seconds.
* Params
* time, int
* Public Methods
* Protected Methods
* Start, void
* Summary
* Initializes the video renderer and other GUI elements.
* FixedUpdate, void
* Summary
* Check the state of the video player state machine.
* Private Methods

**VideoSlider**

* Summary
* Control logic for the video player when the time slider is interacted with.
* Parent Class
* [ClickableObject](#ClickableObject)
* Interfaces
* Constants
* Properties
* Fields
* console, public [VideoPlayerConsole](#VideoPlayerConsole)
* timeLine, public Silder
* prevState, private [EVideoState](#EVideoState)
* Static Members
* Constructor
* Static Methods
* Public Methods
* OnPointerDown, override void
* Summary
* Change the video player state to seeking
* Params
* ptrData, PointerEventData
* OnPointerUp, override void
* Summary
* Update the current frame and set the video player back to the previous state.
* Params
* ptrData, PointerEventData
* OnPointerEnter, override void
* Summary
* Override to stop base method call.
* Params
* ptrData, PointerEventData
* OnPointerExit, override void
* Summary
* Override to stop base method call.
* Params
* ptrData, PointerEventData
* Protected Methods
* Private Methods