GEMS Overview - Table of Contents

Graphical Environment Management System Overview

Travis L. Seymour, PhD 2019

Goal

Create interactive virtual environments defined by a series of static images and various interactions and dynamics.

GEMSedit

The editor allows one to create or edit GEMS environments. Environments are made of a ***views***, which is a viewpoint onto anything that can be depicted. E.g, a view might depict a doorway, or what it looks like inside a room. It might also depict a closeup of an object, or a depiction of what it's like inside that object. It can literally be anything that one can create an image of, including a bunch of text if you want – although that might not be too exciting. To define an environment, photograph, draw, or otherwise create or obtain an image that corresponds to all the views you want to exist within the environment and stitch them together using GEMSedit. Another important thing one does with GEMSedit is to define ***objects,*** which are rectangular sub-regions of view images. Although typically, objects specify meaningful objects that are depicted in the view images (e.g., a book, a cup, or a door), objects may also define areas or sub-areas of a view image not necessarily tied to specific physical objects within that view (e.g., a doorway, a hallway, the left side of a room, or the top of a painting). Last, but not least, GEMSedit is used to definite various environmental actions, interactions, and consequences of those actions or interactions.

Actions. Actions are events that can occur in the environment as a result of some action and optionally if some condition is met. For example, you could specify that a sound effect (the ***action***) if the user left-clicks (the ***trigger***) the mouse on a previously defined object (the ***target object***). Some actions are environmental, e.g., the passage of a certain amount of time, and others are based on user actions (keyboard presses or mouse actions). Although user actions are typically with regard to user-defined objects within some view, there are special navigational actions as well. For example, you can define what happens when the user clicks the left edge of the screen, e.g., you might be taken to a new view. Objects with actions that lead to a view change are called ***portals***, and are typically in regard to some user defined objects. But the left and right side of the screen are special implicit objects one can design as triggers for some actions. Although they are customarily used to allow the users to turn left or right within some implied space, they don't have to be used this way.

Interactions. Interactions are really just normal actions that trigger as a result of one object being dragged (using the mouse left-button) onto another object. This is a great way to increase the richness of actions in the environment. For example, one could drag a key object onto a locked door. This door could be specified to open or portal to another view if clicked on **after** it has had the key dropped onto it. Of course, this would require the environment to be able to keep track of such occurrences. This is one of the uses for ***variables*** within GEMS. An interaction like dragging a key onto a door might result in setting the value of a new variable called "door_unlocked" to True. Clicking the door could be set up to open (i.e, portal to another view) if and only if there exists a variable called "door_unlocked" that currently has the value True.

All actions have 3 required parts and 1 optional parts:

- **Trigger**. These define the event (environmental or user-action) that is required before an action will fire. An example of one environmental trigger is **TotalTimePassed**(*Seconds*) : This trigger fires when at least <*Seconds*> seconds has passed since the current GEMS environment was started. An example of a user-action trigger is **MouseClick**() : This trigger fires whenever the mouse is left-clicked.
- **Action**. These define the action that occurs when a trigger occurs. For example, a common action portals the user to another view: **PortalTo**(*View*) : This action causes GEMS to load another view with the view ID of <*View*>. If there is a working internet connection, you can cause text to be spoken with this action: **SayText**(*Message*) : This action causes GEMS to speak the given <*Message*> using the Google's text-to-speech API.
- **Enabled**. This is either the value *True* or *False*. Setting this to *False*, allows an action to stay defined but not activated.

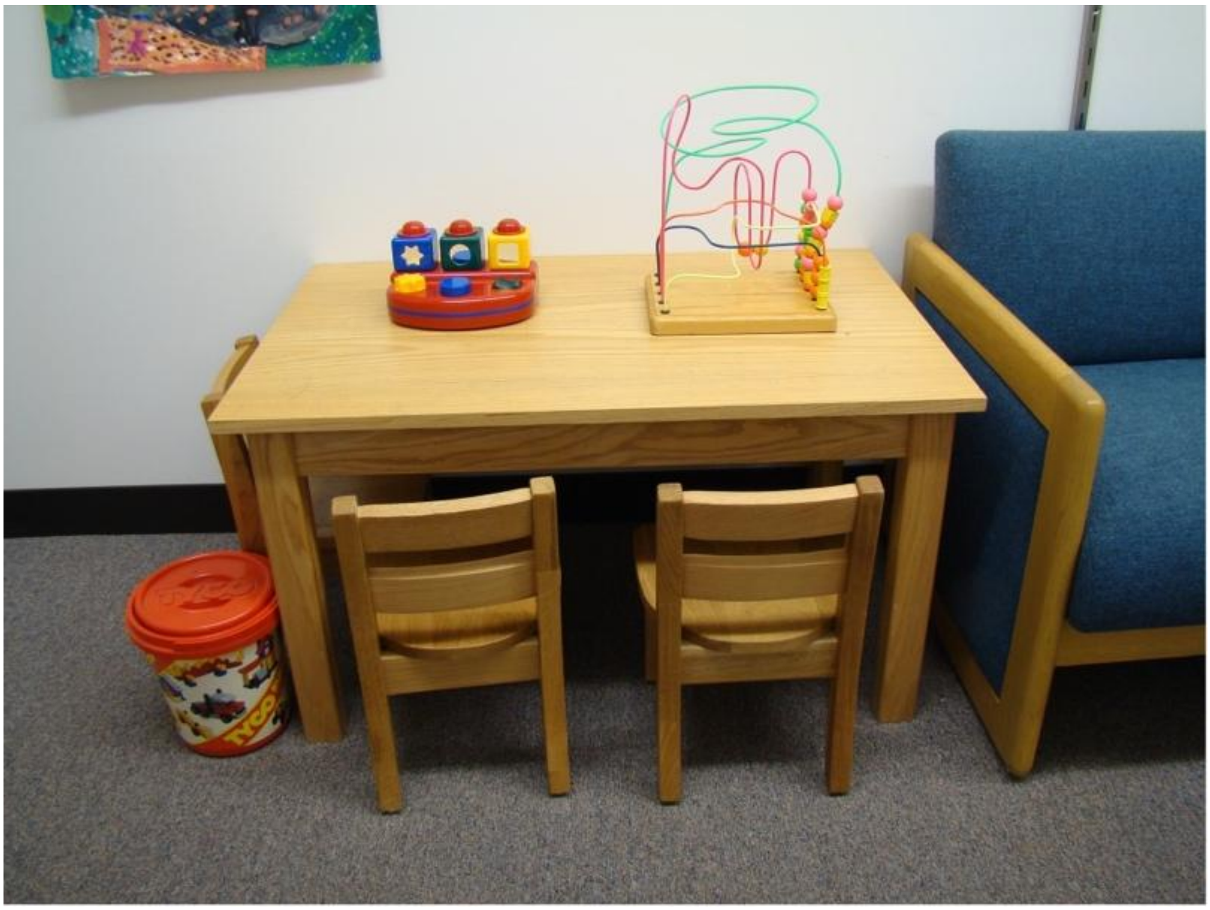
Actions can also have conditions, which are environmental states that must be met in order for the action to be considered. Conditions are optional, so if no condition is defined, then only the action's trigger determines when it activates. One common condition is **ViewTimePassed**(*Seconds*) : This condition returns *True* if at least <*Seconds*> seconds has passed since the current view was displayed. Another common condition is **VarValueIs**(*Variable*, *Value*) : This condition returns True if the user created token <*Variable*> exists and currently has the value <*Value*>.

GEMS actions and interactions can be specified at various levels.

Environment Actions. Sometimes it is useful to specify some conditions, triggers, and actions at the environmental level. These are actions that occur no matter which view is currently active. There are only 2 triggers at this level: **TotalTimePassed**(*Seconds*) : This condition returns True if at least <*Seconds*> seconds has passed since the current GEMS environment was started, and **KeyPress**(Key) : This trigger fires when <*Key*> is entered on the keyboard.

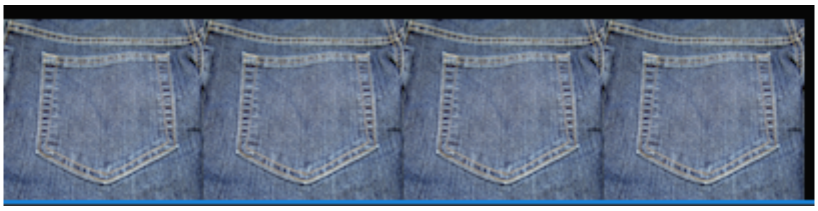
View and Object Actions. There are many actions that can be defined in GEMS. Technically, the only real view-only action is one that triggers on the **ViewTimePassed**. Otherwise, all actions are essentially based on object-level actions and interactions. You can see a list of all the conditions, actions, and interactions in the appendix.

What you can do with objects in GEMS. Remember that an object in GEMS is just a rectangular region that the designer had designated as such. There is no obligation to define everything in a view that is visible as an object, but doing so allows some trigger (clicking, hovering, dragging, etc.) relative to that object to have an effect (play a song, portal to another view, speak some text, make some object disappear). When you create a view, you have to at least specify a **foreground picture** and a **background picture**. In some cases, these will be the same image. However, if they are not the same, interesting possibilities arise. For example, consider these foreground and background images:

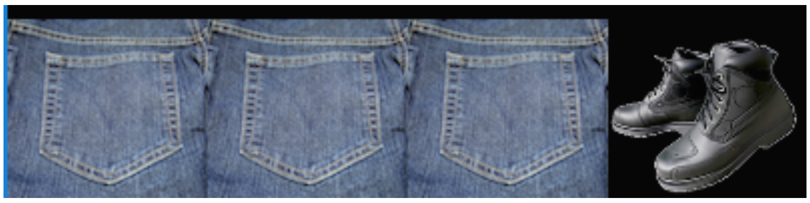


If the designer specifies a foreground image that has some objects depicted and a background image that is identical, except that those objects are missing, then in the environment, several things become possible. One is that the object can be specified as visible or hidden and be revealed or hidden later as a result of some action or interaction. In addition, if an object defined this way is dragged with the mouse, it will appear as if it has been picked up and moved across the screen. This illusion is created because when the object is dragged, the location in which it was defined is quickly replaced with the same region from the background. Note that for an object to be “*takeable*”, it has to be specifically specified as such in GEMSedit. Otherwise, users won’t be able to move it with the mouse. However, it could still be hidden or revealed. Another powerful that that results from using the on/off design for foreground and background images and specifying an object as takeable is that it can be dragged a) onto other objects, and b) into your pockets.

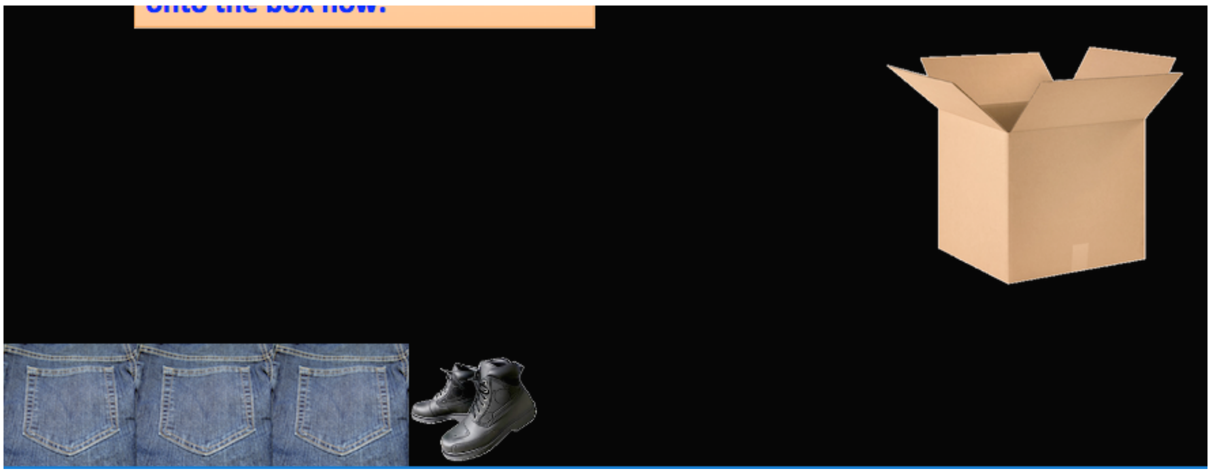
One of the many triggers available for objects is what happens if a specific other object is dragged onto it. Making an object takeable allows the designer to create interesting interactive possibilities. For example, you might specify that if a toy is dragged onto a bin, it disappears from the table. Thus, allowing users to clean up organize objects. Programmatically, the object has just be marked as non-visible as a result of an interaction type action. At the environment level, several global options can be specified. One is how many (including 0) pockets will be available. Pockets are icons that show up in the bottom left corner of the display that look like denim jeans back pockets, e.g.:

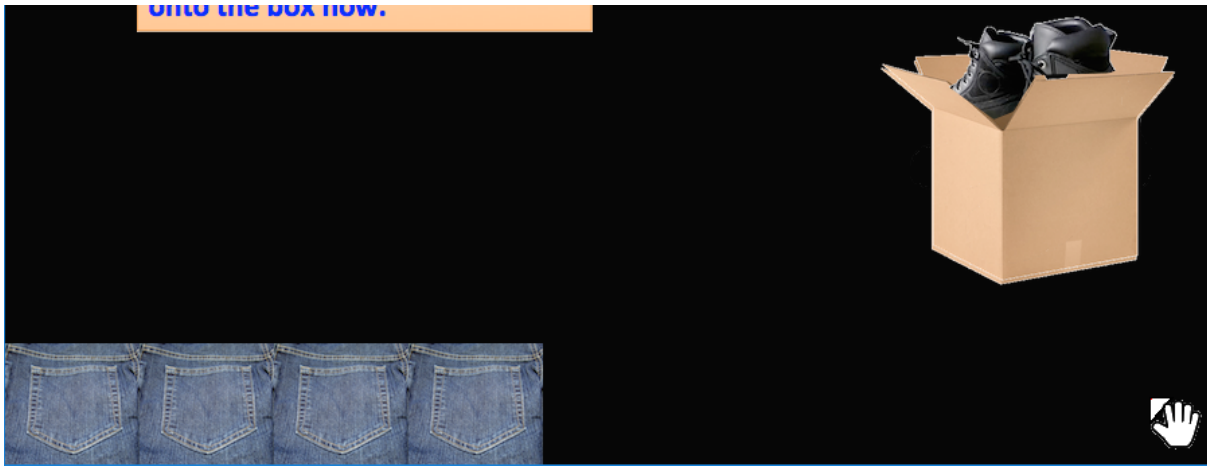


If a takeable object is dragged to a pocket, it will appear as if it is dragged out of the environment and into the pocket:

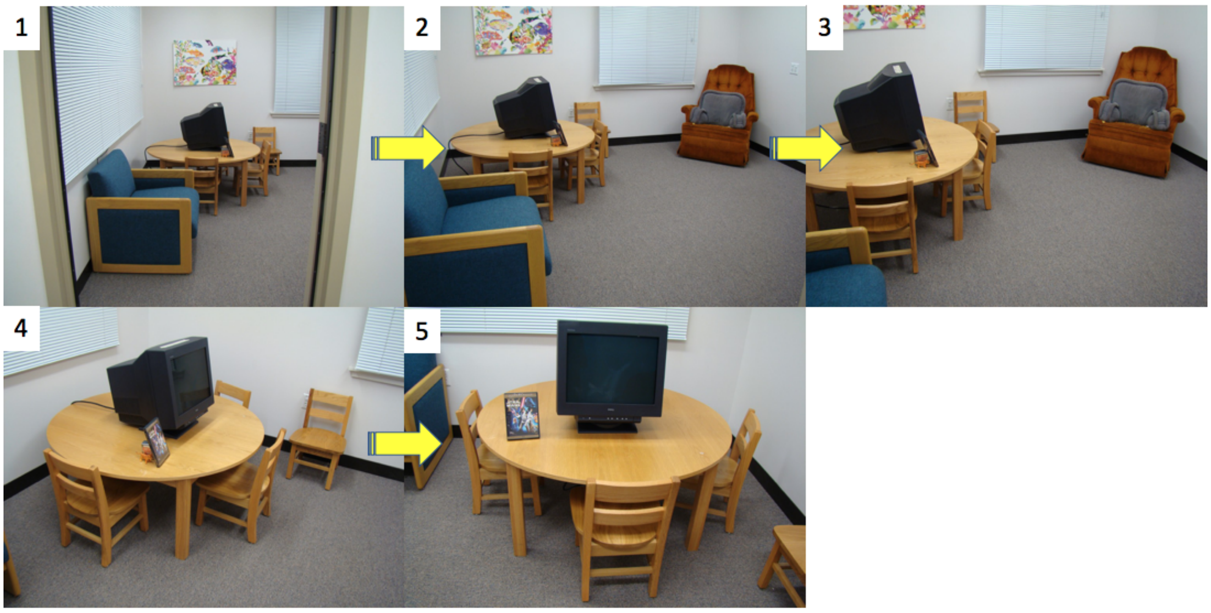


Pockets allow users to collect objects and move them from one view to another. Right-clicking a pocketed object will cause it to return to its original location. If you wanted the user to drag the boots into a box depicted in a view, you couldn't change the foreground picture so that it looked like the boots were now in it. However, you could do one of 2 things: One you could have a depiction of the boots in the box in the background image and (critically) within the same exact rectangular space, have a version of the box without the shoes in the foreground image. This way, dragging the shoes onto the empty box could be made to result in the box object disappearing, revealing the boots-in-box region on the background. One possible action in GEMS is to place an image from the computer as an overlay of sorts on top of the foreground image. Thus, another approach would be to specify that if the boots are dragged onto the box object, that area could be covered by a small image just depicting the boots in the box. Either way, the effect for the user looks as if they have dragged the boots out of their pocket and into the box, e.g.:

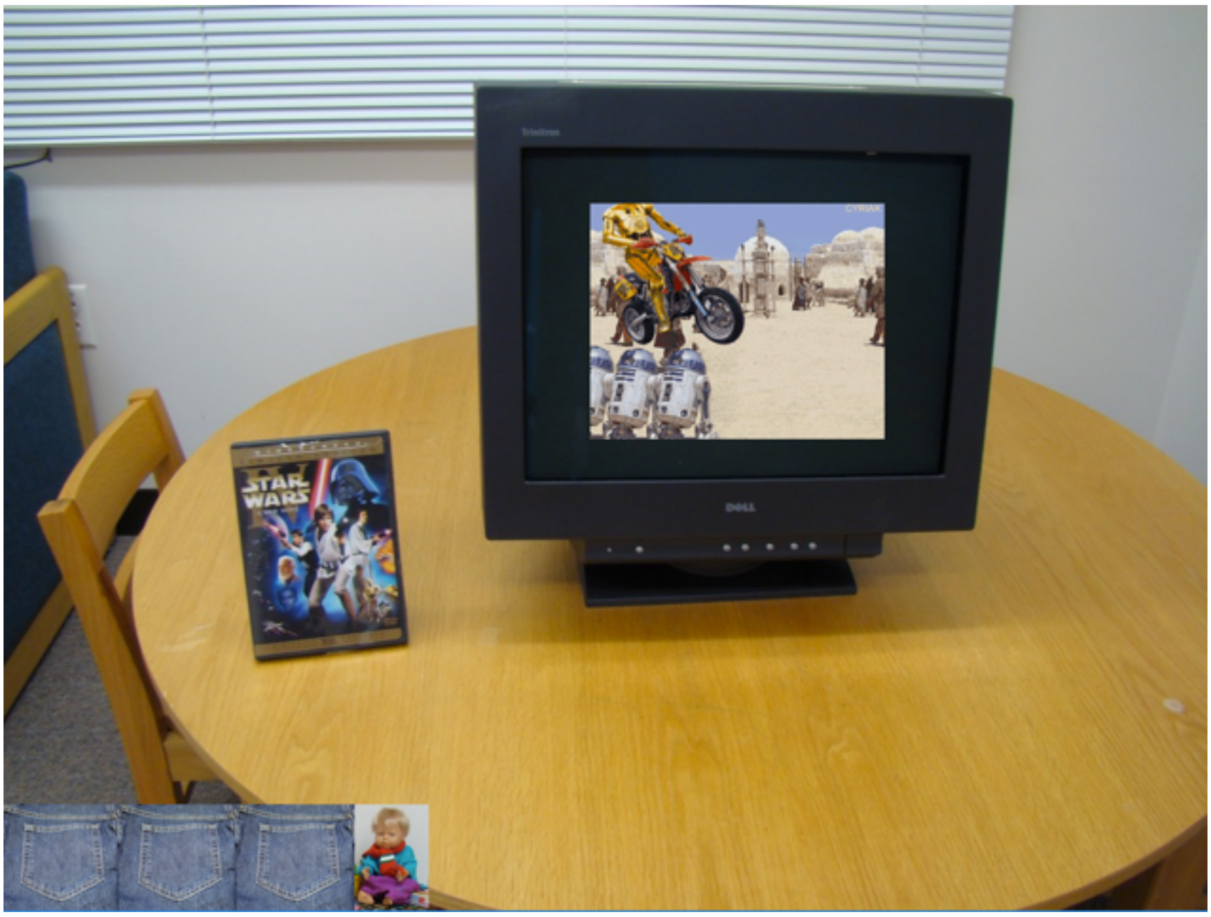




Automatically Portaling. Most often, moving from one view to another happens a result of some user-initiated action or interaction. However, portaling can also occur as a result of global timed events, or view-level timed events. One use for global timed portals may be to institute a deadline for interaction with a GEMS environment. For example, maybe you want to take the user to a view with the words TIME'S UP after 30 minutes. View level timers can be used to make things happen relative to entering a view. For example, maybe after 2 minutes, if some action has not been taken by the user, you could show a textbox containing a clue, or portal to another view. Sometimes we use view-level timed portaling to quickly move the user through a sequence of views to establish a crude animation. This can be used to give the user the sense of moving over a long distance, or be used to establish the layout of a space. If, each of several views contained a **ViewTimePassed** trigger set to go off .25 seconds after the view is entered that fires a **PortalTo** action, then you could fly the user from one point in a room to another. E.g.:



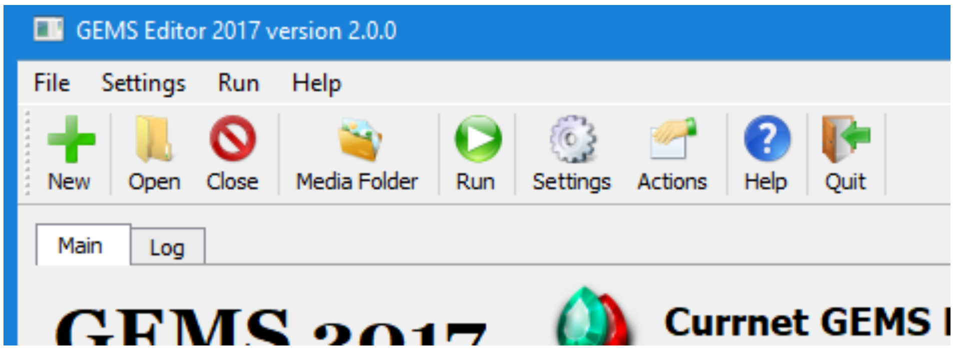
Left-clicking on view #1 could automatically portal you through the subsequent views until you stopped at view 5. This is animated in a way that feels as if you are flying from the door, around the room, to the table.

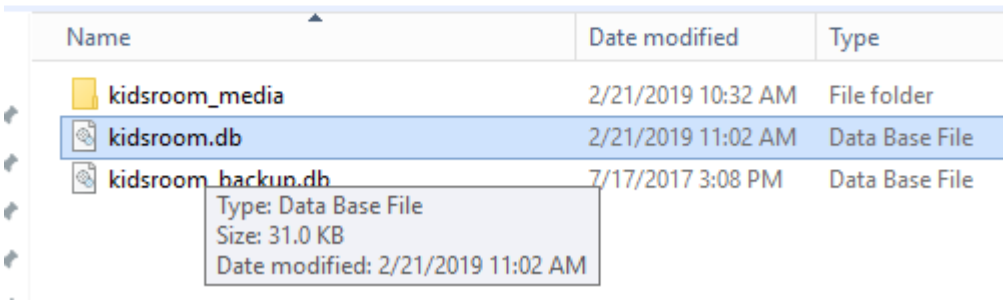


In the test environment depicted above, an interaction is defined in view 5. If the dvd cover (which is not takeable in this case – we forgot to take a picture without the dvd) is dragged onto the monitor, a little video plays. Note that although the dvd is not takeable it is still ***draggable***. The downside of this arrangement is that the dvd will not look as if it is being dragged, but the interaction will still work.

Using GEMSedit

Loading an existing GEMS Environment. Gems environments consist of a folder that contains an environment database file (e.g., kidsroom.db) and a folder of media (pictures and audio files), named [ENVNAME]_media, e.g., kidsroom_media. To open an environment, press the OPEN ICON, navigate to an environment folder, and choose the database file.

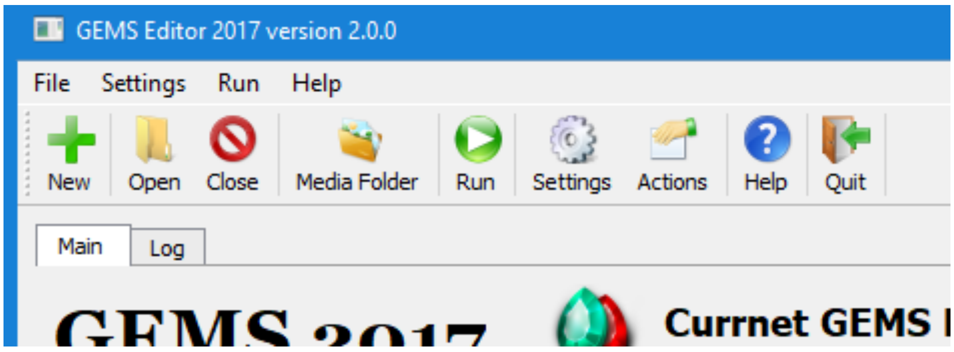

GEMS Editor 2017 version 2.0.0
File Settings Run Help
New Open Close Media Folder Run Settings Actions Help Quit
Main Log
GEMS 2017
Currnet GEMS

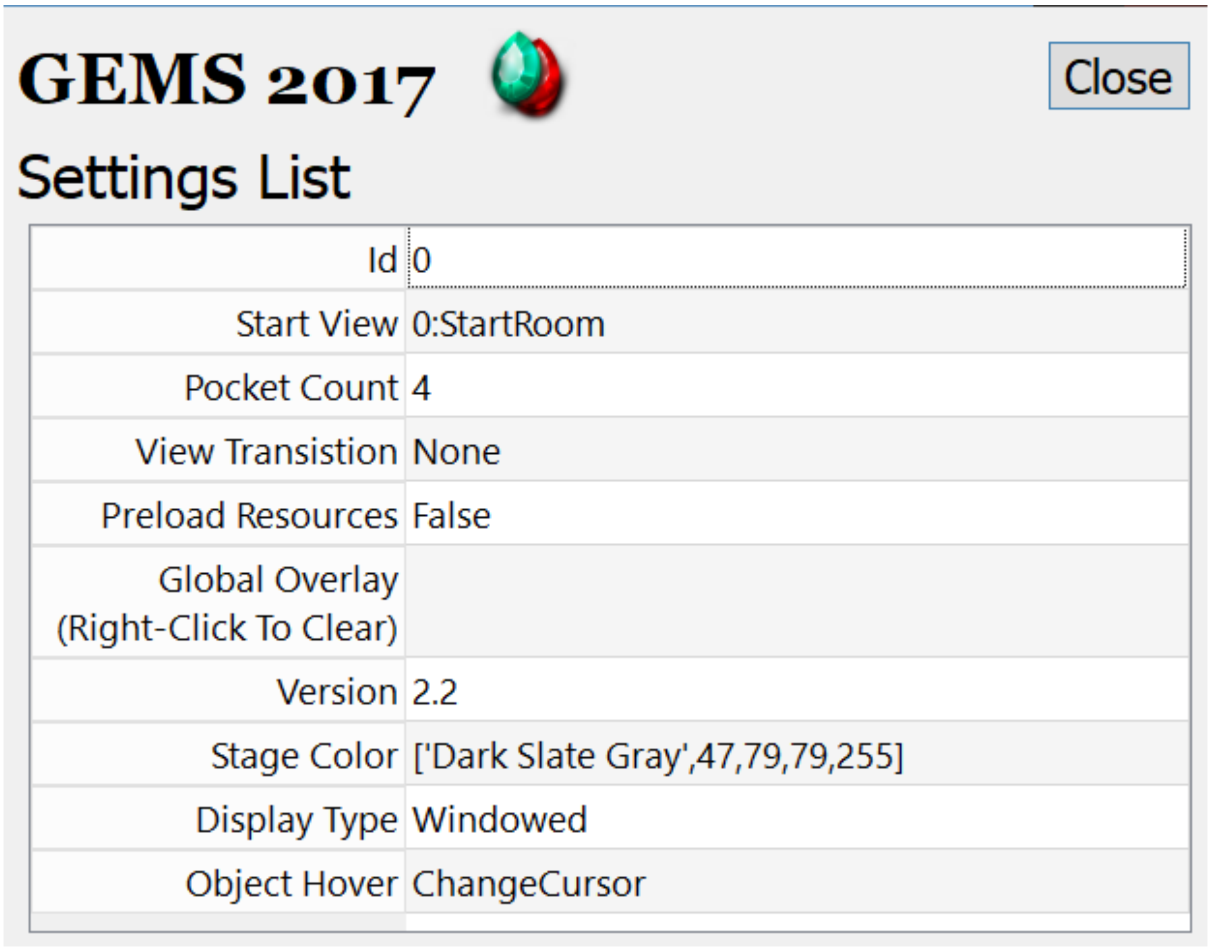


Name	Date modified	Type
kidsroom_media	2/21/2019 10:32 AM	File folder
kidsroom.db	2/21/2019 11:02 AM	Data Base File
kidsroom_backup.db	7/17/2017 3:08 PM	Data Base File

Type: Data Base File
Size: 31.0 KB
Date modified: 2/21/2019 11:02 AM

Global Environment Settings. There are several global settings.


GEMS Editor 2017 version 2.0.0
File Settings Run Help
New Open Close Media Folder Run Settings Actions Help Quit
Main Log
GEMS 2017
Currnet GEMS



Id	0
Start View	0:StartRoom
Pocket Count	4
View Transistion	None
Preload Resources	False
Global Overlay (Right-Click To Clear)	
Version	2.2
Stage Color	['Dark Slate Gray',47,79,79,255]
Display Type	Windowed
Object Hover	ChangeCursor

ID: Hmm...I don't know what this is?!

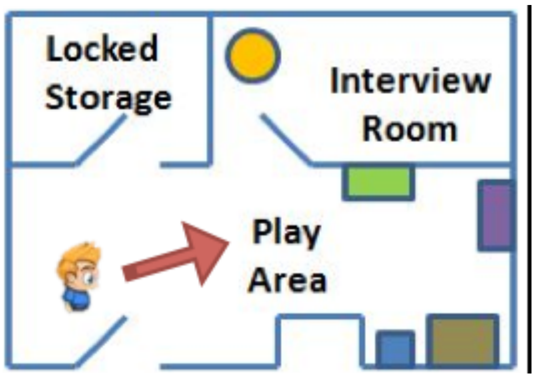
Start View: This defines the first view that will be loaded when the environment starts. You change this to any existing view in the environment.

Pocket Count: The number of pockets to feature in the environment

View Transition: This allows you to specify how GEMS transitions from one view to the other. For now, this does nothing – views transition via jump. Later, I hope to add a Fade transition and possibly others.

Preload Resources: This is a currently non-implemented feature to load all project media into memory for fast environment navigation. At the moment this is not practical and has been disabled no matter what is specified here.

Global Overlay: If a picture is specified here, it will be overlayed onto the upper left corner of the environment for ***every*** view. This cannot be changed while the environment is running and may have limited use cases. For a flexible overlay, you'll notice that when designing view, you can specify a view-level overlap that can be different from one view to the next. We've used this for a dynamically updating map of where one is in an environment. E.g.:



Version: This field will go away soon. Ignore it.

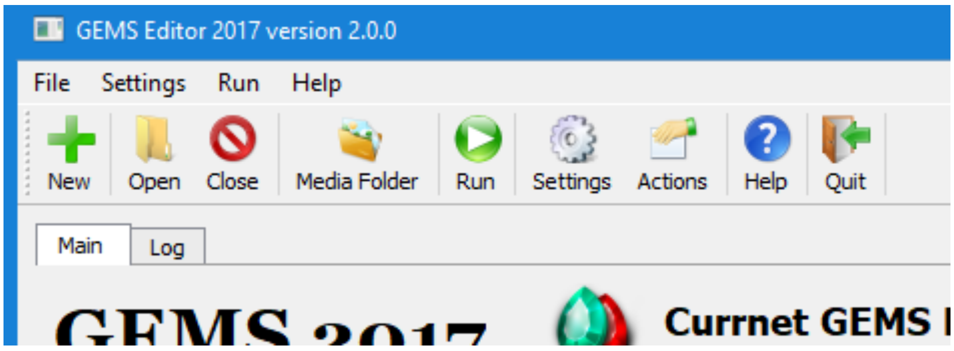
Stage Color: Hopefully you won't have to deal with this. The size of the environment is dictated by the **StartView's foreground picture**. Any subsequent view that uses foreground/background smaller than this will result in a colored background showing the extent of the area not used. This setting specifies the color of that region. Make all of your view images the same size to avoid this situation!

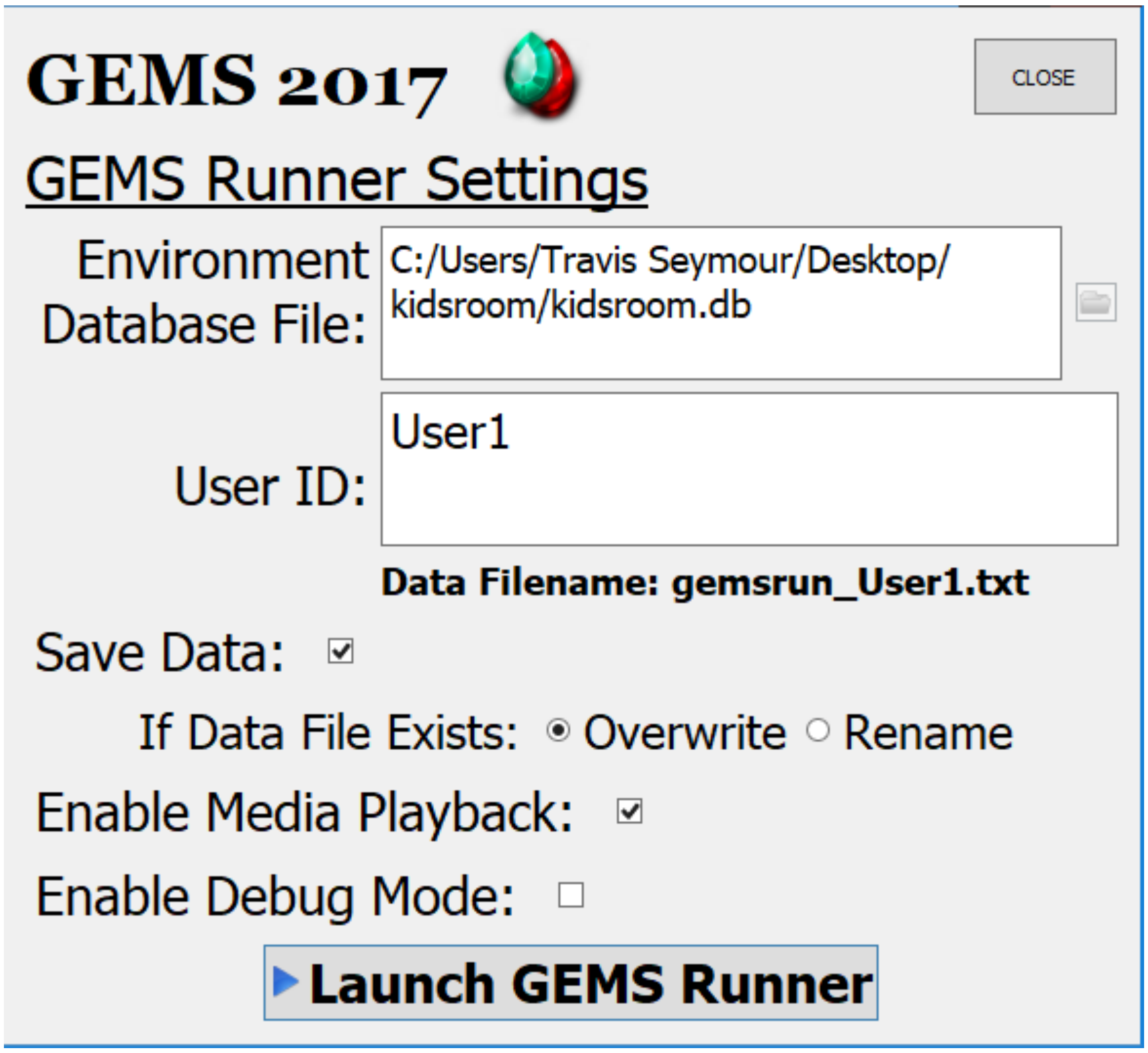
Display Type: Can be either Windowed or FullScreen. Windowed mode is nice for designing because one will also need to manage media files. However, for a more immersive experience, switch to FullScreen. Note: The most recent version of GEMS needs significant testing in FullScreen mode – it has mostly been set to Windowed for development and testing.

Object Hover: This dictates what happens when someone hovers over a region defined as an object. The options are ChangeCursor (the cursor switches from white to yellow alerting the user that this object is defined), FrameObject (draws a yellow rectangle around the boundaries of the object), and ShowName (puts the objects name in yellow at the bottom left of its bounding box). The Object Hover option also allows combinations such as Frame+Cursor, Name+Cursor, and Cursor+Frame+Name. Most of these permutations were created for designing environments, but ChangeCursor may be a useful user facing ui choice.

Running a GEMS Environment.

To run a loaded environment, just click on the run icon:





Environment Database File: C:/Users/Travis Seymour/Desktop/kidsroom/kidsroom.db

User ID: User1

Data Filename: gemsrun_User1.txt

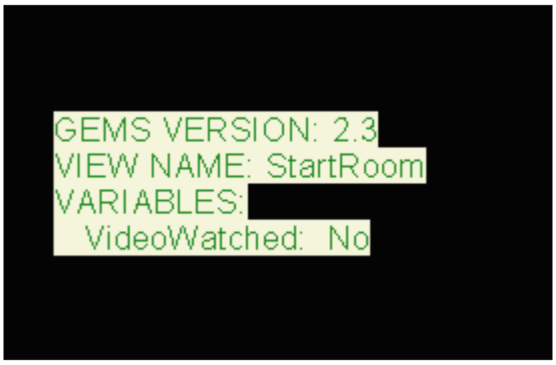
Save Data: ☑

If Data File Exists: [✓ Overwrite] Rename

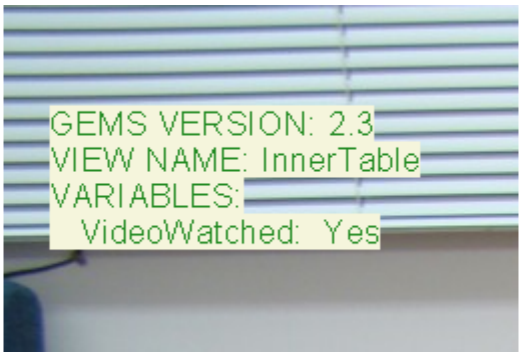
Enable Media Playback: ☑

Enable Debug Mode: ☐

The **Environment Database File** is not editable and will be set to whatever environment is currently loaded into the editor. GEMS is designed to save data about everything that happens during a run, a user id can be used to distinguish data from multiple users. If this is left at its default, it will always use “User1” as the user id. You can toggle whether to actually save the data and whether to overwrite or rename an existing data file for a particular user. Media playback can be disabled when quiet operation is desired. Debug mode use helpful during environment design because it shows the current status of any defined environment variables. E.g.:



```
GEMS VERSION: 2.3
VIEW NAME: StartRoom
VARIABLES:
  VideoWatched:  No
```



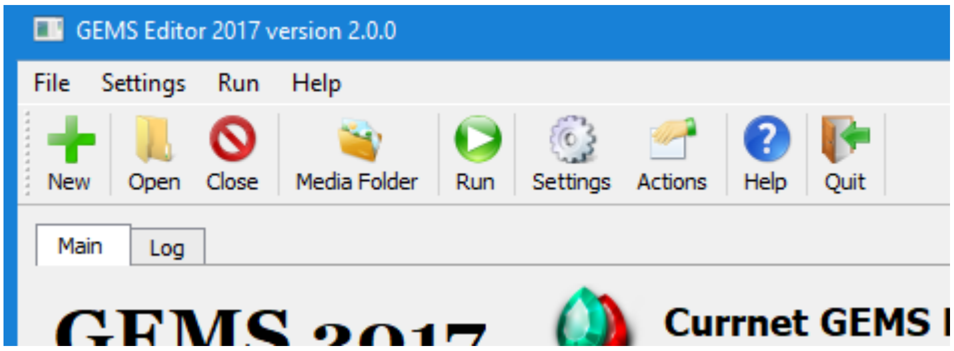
```
GEMS VERSION: 2.3
VIEW NAME: InnerTable
VARIABLES:
  VideoWatched:  Yes
```

This information floats in the upper left hand corner when Debug Mode is enabled.

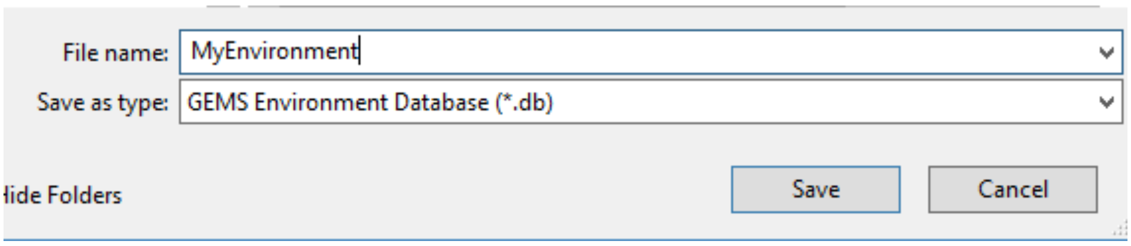
Finally, press the Launch GEMS Runner button to start. To stop, press ESCAPE (e.g., in FullScreen Mode) or click the red close-window icon (e.g., in Windowed Mode).

Creating a New GEMS Environment.

1. Press the NEW button



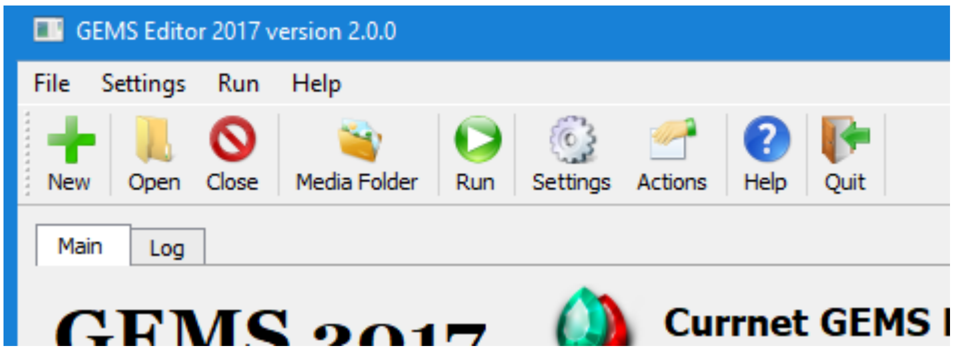
2. Navigate to where you want the environment folder to be created (e.g., Desktop) and type in a name of your project (no extension). For example, enter “MyEnvironment” and press **SAVE**.


File name: MyEnvironment

Save as type: GEMS Environment Database (*.db)

3. You will notice that GEMS creates a new environment with 2 default views (StartRoom and EndRoom). A couple of objects and interactions have been defined. You can delete these and specify your own environment.

4. Click the **MediaFolder** icon to open your media folder.



5. Do not bother the files already here. Some of them are needed for GEMS operation. Dump all the images and sound files (e.g., .png or .jpg image files, and .mp3 or .ogg audio files. Many formats are supported). You will need for your environment here. Then you will be ready to stitch them all together to form your environment.

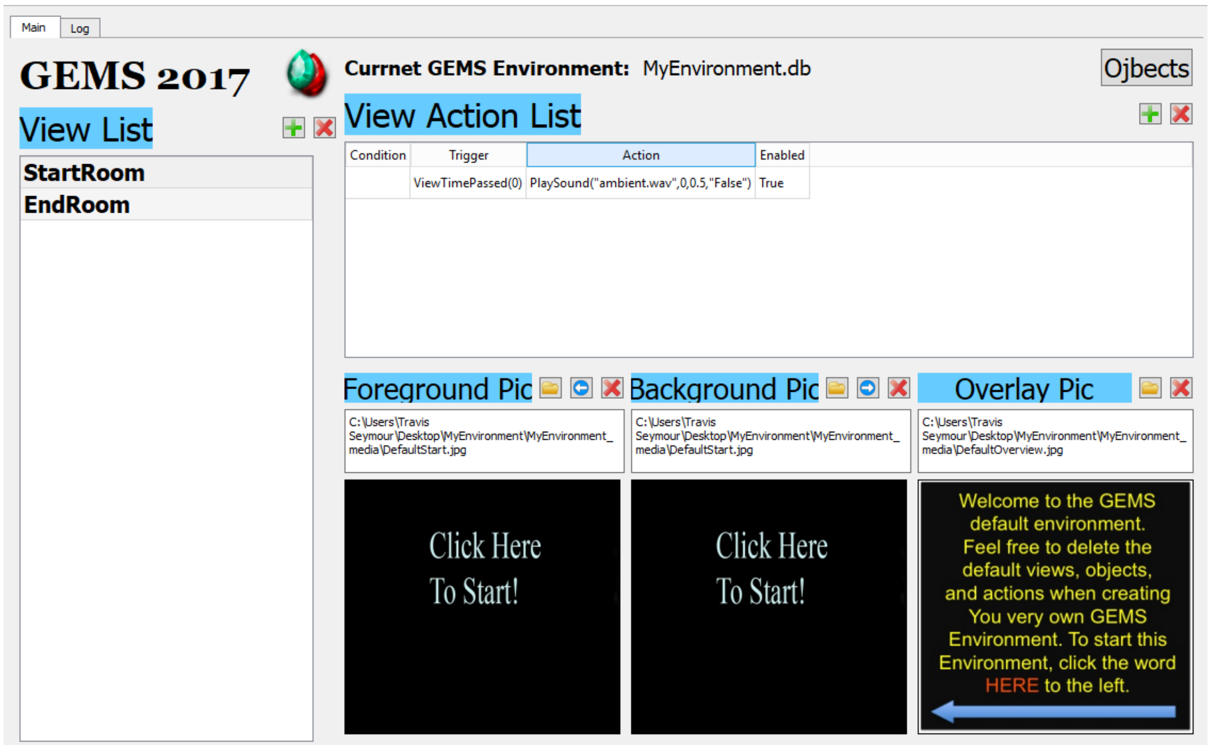
6. I suggest you plan out your environment on paper first. Which views will you use and to which other views could a user portal from each view.

7. Next it's time to take pictures. Use a tripod and try to have as little difference as possible (i.e., no difference) between the camera position when taking foreground and background images for a particular view. You want to avoid lighting differences as well – keep track of how your own location in the room alters the light. Planning is key, if you have to go back later to get another shot, the lighting may be very different.

8. If you already know which sounds you want, obtain them or record them to .wav or .mp3 and copy them to your media folder as well. Note: All pictures and sound files go into the root of your media folder. Don't create any sub-folders.

9. Once you have a plan, and your media, you're ready to edit your environment.

Defining and Editing Views



Curnet GEMS Environment: MyEnvironment.db

StartRoom
EndRoom

Condition	Trigger	Action	Enabled
	ViewTimePassed(0)	PlaySound("ambient.wav",0,0.5,"False")	True

C:\Users\Travis Seymour\Desktop\MyEnvironment\MyEnvironment_media\DefaultStart.jpg

C:\Users\Travis Seymour\Desktop\MyEnvironment\MyEnvironment_media\DefaultStart.jpg

C:\Users\Travis Seymour\Desktop\MyEnvironment\MyEnvironment_media\DefaultOverview.jpg

Welcome to the GEMS default environment. Feel free to delete the default views, objects, and actions when creating You very own GEMS Environment. To start this Environment, click the word HERE to the left.

1. These icons allow you to create a new empty view, or delete the currently selected view in the list, respectively. Be careful with deleting views – this cannot be undone. Also make sure you have selected the view you really want to delete. Adding a new view requires only a new view name. It is possible to have multiple views with the same name – we do not recommend this.

2. The folder icon lets you specify a foreground picture from those in your media folder. The arrow icon copies the value from the current background picture. The red X deletes the current picture entry (this does not affect the image file on disk).

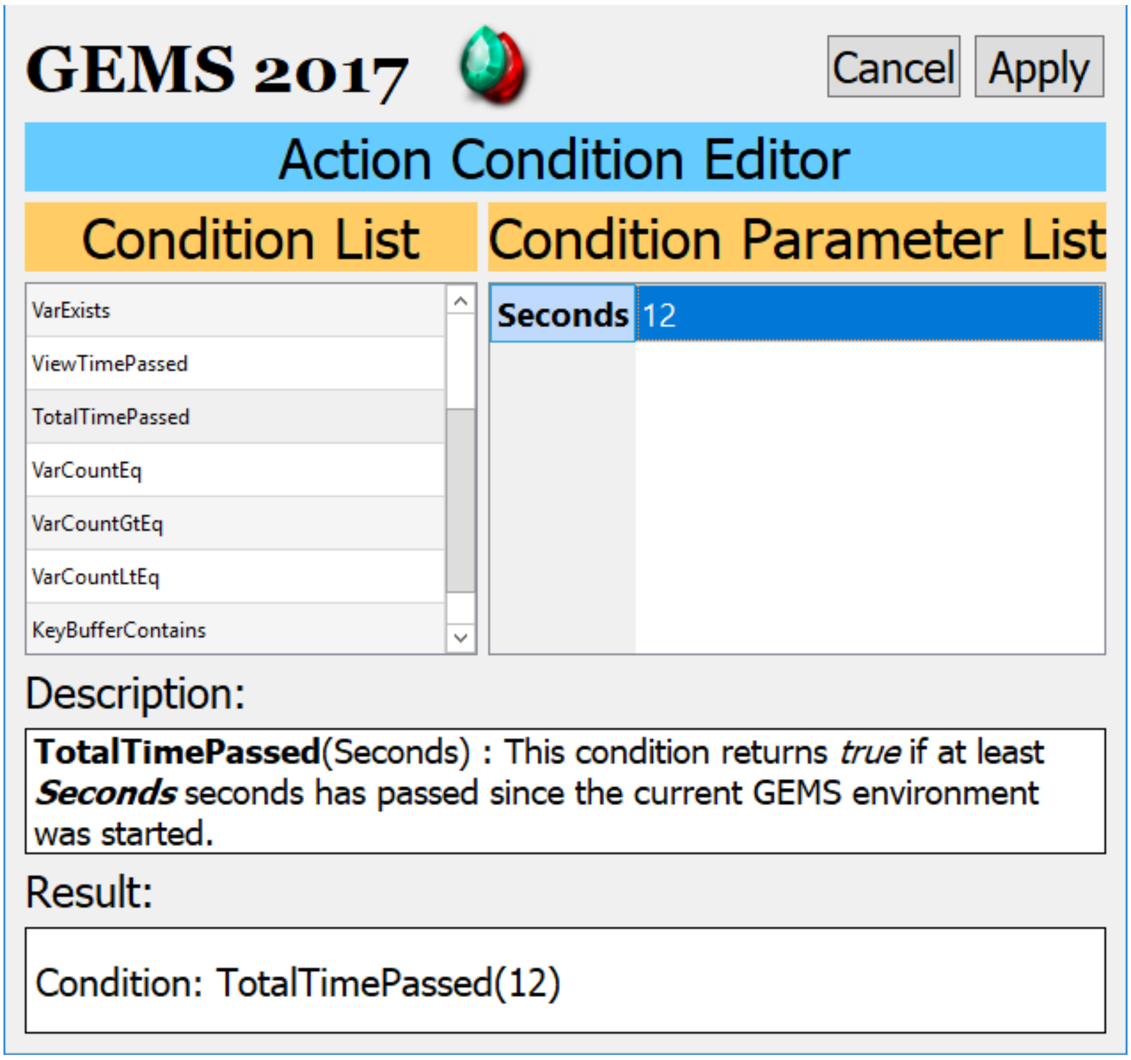
3. Same as #2, but for the background picture. The blue arrow will make the current background the same image currently used for the foreground. **Please ensure that all view foreground and background pictures the exact same dimensions or odd things can result. In a few use cases, this might be desirable, but not usually.**

4. Select or delete an overlay picture. This picture will be displayed relative to the upper right corner of the view. This overlay could be as large as the foreground and background pictures, but for many use cases, it will be somewhat smaller. E.g., a map, or special message.

5. All the windows on this row show a depiction of which image is currently selected. If you click on this window, you will be able to view a full size version. Hit the ENTER key to return.

6. Click the green plus to create a new empty action. All cells will initially turn red to indicated that now enough information has been supplied for it to actually work. This will change when you're entered at least a trigger, action (the condition is optional). The red X will permanently delete the currently selected action. Please be careful that you have selected the appropriate view action and that you really want to delete it. This cannot be undone. It may be wiser to disable an action than to delete it if you think you may want to use re-enable it later.

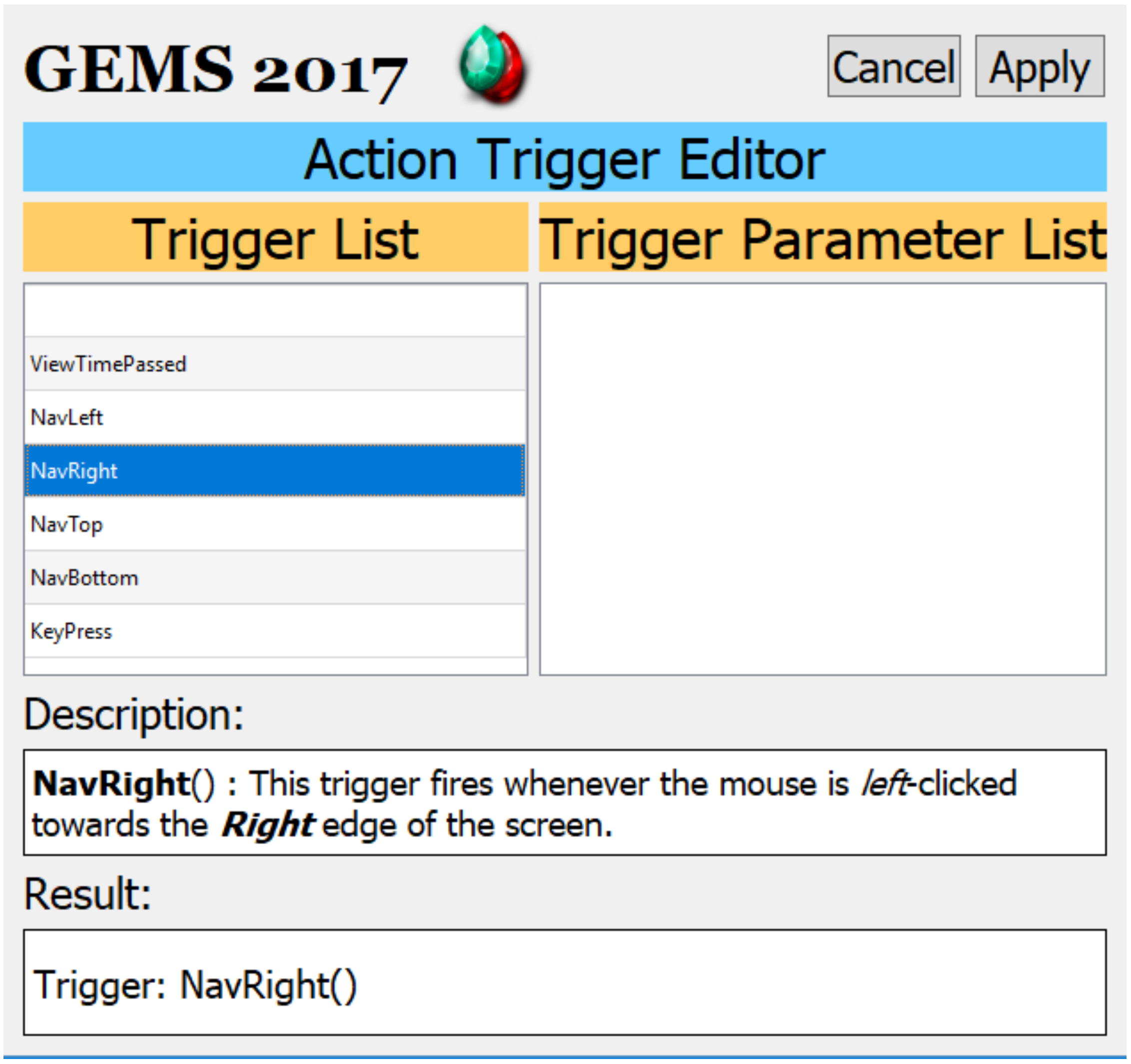
7. Clicking on either the Condition, Trigger, Action, or Enabled cells allows you to specify their contents for that action. This occurs via dialogs that pop up when you click each cell. For example, clicking the Condition, Trigger, and Action cells will pop up dialogs like these:



Seconds 12

TotalTimePassed(Seconds) : This condition returns *true* if at least ***Seconds*** seconds has passed since the current GEMS environment was started.

Condition: TotalTimePassed(12)

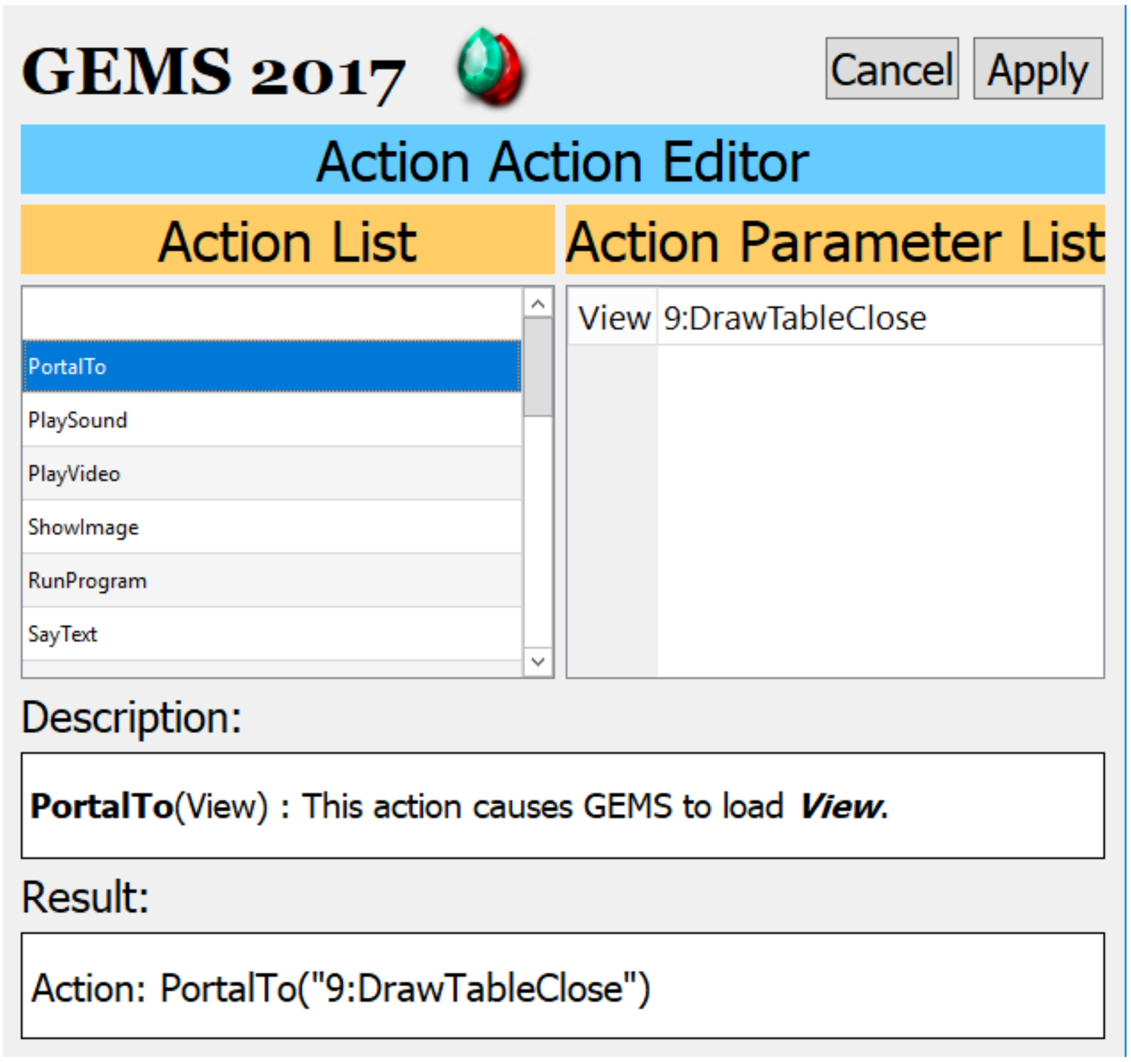


Description:

NavRight() : This trigger fires whenever the mouse is *left*-clicked towards the ***Right*** edge of the screen.

Result:

Trigger: NavRight()



View	9:DrawTableClose

Description:

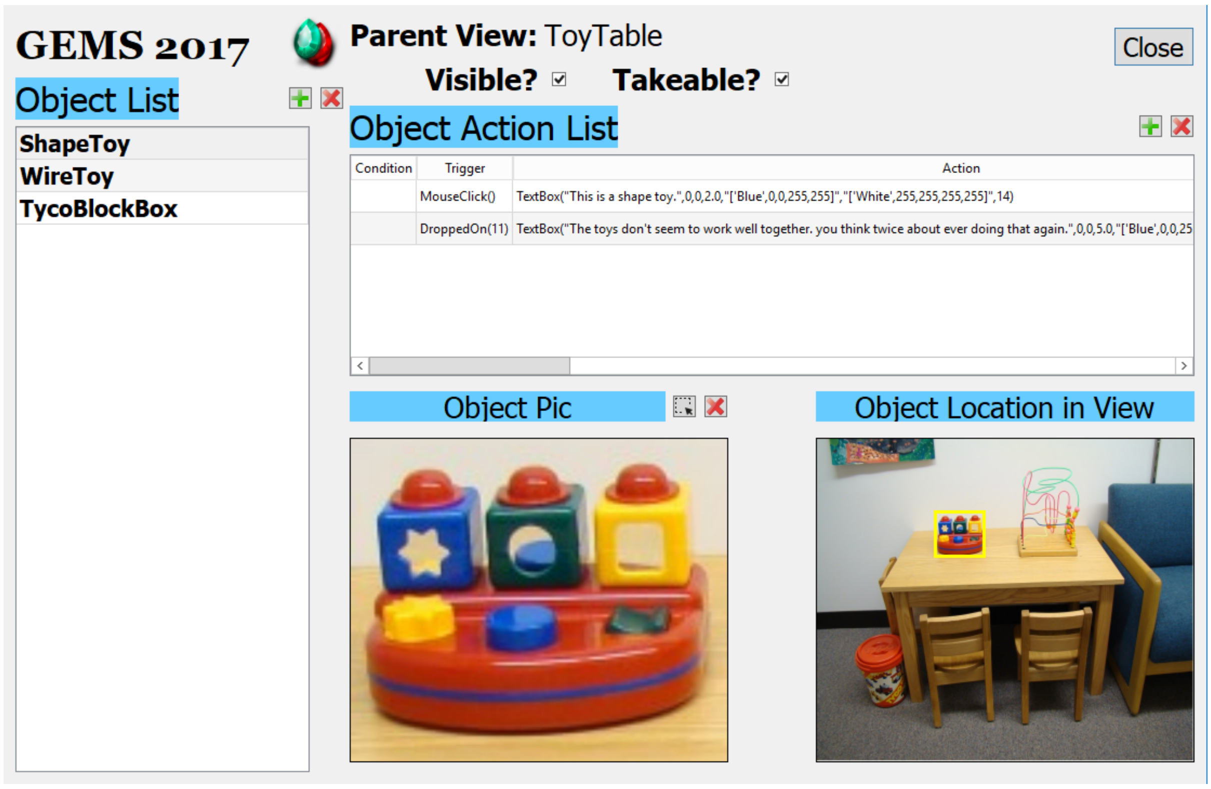
PortalTo(View) : This action causes GEMS to load ***View***.

Result:

Action: PortalTo("9:DrawTableClose")

8. This button is used to open a new screen that allows the design and specification of objects for this view. Make sure you have specified your foreground and background pictures before creating objects.

Defining and Editing View Objects



ShapeToy
WireToy
TycoBlockBox

Condition	Trigger	Action
	MouseClick()	TextBox("This is a shape toy.",0,0,2.0,"['Blue',0,0,255,255]","['White',255,255,255,255]",14)
	DroppedOn(11)	TextBox("The toys don't seem to work well together. you think twice about ever doing that again.",0,0,5.0,"['Blue',0,0,25

1. Similar to with the view editor, these buttons allow you to add define new objects, or remove previous object definitions. Be careful, deletions cannot be undone.

2. Objects are simply sub-regions of your existing foreground image. So clicking the lasso tool here allows you to define your object as a rectangular. The two image regions below #2 and to the right of it (Object Location View) show you the selected region in context and in a magnified fashion. Pressing the red X button will allow you to clear out the object picture selection.

3. This checkbox determines whether this object will be marked as visible, and thus whether it will show up when the environment is run. Actions and interactions can result in this being toggled.

4. "Takeable" refers to whether or not this object is meant to be removed from its default location and potentially dragged to the pockets for transport across views. If you define an object as takeable, you really should make sure that it is not visible in the background image. This will lead to the most natural effect when users drag it out of position within the environment. An exception to this may be a stack of paper where you want the user to appear to have taken a sheet of paper while leaving the stack in place.

5. Just like at the view level, these buttons allow you to add or remove actions.
6. Just like at the view level (and using the same dialogs shown above), this list allows you to manipulate the conditions, triggers, and consequences of each defined action.

Other Thoughts

1. You don't have to save anything...all changes are saved to you environment database immediately.
2. If you want to move around or share your environment, everything will be in your project folder. Essentially your xxxxxx.db file and the xxxxxx_media folder.
3. You may need to learn about basic photo editing to manage images that are too big, or are of different sizes. Same idea for basic audio editing for controlling volume and cropping.

Appendix - GEMS API Reference

This section describes all conditions, triggers, and actions available in GEMS (Graphical Environment Management System).

Restriction Scopes

Actions, conditions, and triggers can be restricted to specific contexts:

Scope	Description
`view`	Can be used at the view level
`object`	Can be used on objects within a view
`global`	Can be used in global scripts
`pocket`	Can be used on pocket objects

Triggers

Triggers define events that initiate script execution.

ViewTimePassed

```
ViewTimePassed(Seconds: float)
```

Description: Fires when the specified number of seconds has elapsed since entering the current view.

Parameters:

- `Seconds` (float): Time in seconds to wait before triggering

Restrictions: view

TotalTimePassed

```
TotalTimePassed(Seconds: float)
```

Description: Fires when the specified number of seconds has elapsed since the environment started running.

Parameters:

- `Seconds` (float): Total elapsed time in seconds

Restrictions: global

MouseClick

```
MouseClick()
```

Description: Fires when the user clicks on an object or pocket.

Parameters: None

Restrictions: object, pocket

NavLeft

```
NavLeft()
```

Description: Fires when the user clicks in the left navigation area of the view.

Parameters: None

Restrictions: view

NavRight

```
NavRight()
```

Description: Fires when the user clicks in the right navigation area of the view.

Parameters: None

Restrictions: view

NavTop

```
NavTop()
```

Description: Fires when the user clicks in the top navigation area of the view.

Parameters: None

Restrictions: view

NavBottom

```
NavBottom()
```

Description: Fires when the user clicks in the bottom navigation area of the view.

Parameters: None

Restrictions: view

DroppedOn

```
DroppedOn(Object: objnum)
```

Description: Fires when a specific object is dropped onto this object or pocket.

Parameters:

- `Object` (objnum): The ID of the object that must be dropped

Restrictions: object, pocket

KeyPress

```
KeyPress(Key: key)
```

Description: Fires when the specified key is pressed.

Parameters:

- `Key` (key): The key to listen for

Restrictions: view, global

Conditions

Conditions are evaluated to determine whether associated actions should execute.

VarValueIs

```
VarValueIs(Variable: varname, Value: value)
```

Description: Returns true if the specified variable exists and equals the given value.

Parameters:

- `Variable` (varname): Name of the variable to check
- `Value` (value): Value to compare against

Restrictions: view, object, global, pocket

VarValueIsNot

```
VarValueIsNot(Variable: varname, Value: value)
```

Description: Returns true if the specified variable does not equal the given value (or doesn't exist).

Parameters:

- `Variable` (varname): Name of the variable to check
- `Value` (value): Value to compare against

Restrictions: view, object, global, pocket

VarExists

```
VarExists(Variable: varname)
```

Description: Returns true if the specified variable exists.

Parameters:

- `Variable` (varname): Name of the variable to check

Restrictions: view, object, global, pocket

ViewTimePassed

```
ViewTimePassed(Seconds: number)
```

Description: Returns true if the specified time has passed since entering the current view.

Parameters:

- `Seconds` (number): Time in seconds

Restrictions: view, object

TotalTimePassed

```
TotalTimePassed(Seconds: number)
```

Description: Returns true if the specified time has passed since the environment started.

Parameters:

- `Seconds` (number): Time in seconds

Restrictions: view, object, global, pocket

VarCountEq

```
VarCountEq(Count: number)
```

Description: Returns true if the total number of variables equals the specified count.

Parameters:

- `Count` (number): Expected variable count

Restrictions: view, object, global, pocket

VarCountGtEq

```
VarCountGtEq(Count: number)
```

Description: Returns true if the total number of variables is greater than or equal to the specified count.

Parameters:

- `Count` (number): Minimum variable count

Restrictions: view, object, global, pocket

VarCountLtEq

```
VarCountLtEq(Count: number)
```

Description: Returns true if the total number of variables is less than or equal to the specified count.

Parameters:

- `Count` (number): Maximum variable count

Restrictions: view, object, global, pocket

KeyBufferContains

```
KeyBufferContains(Keys: value, IgnoreCase: bool)
```

Description: Returns true if the key buffer contains the specified key sequence.

Parameters:

- `Keys` (value): Key sequence to search for
- `IgnoreCase` (bool): If true, comparison is case-insensitive

Restrictions: view, global

Actions

Actions are commands that modify the environment state or trigger effects.

PortalTo

```
PortalTo(View: viewnum, VidFile: vidfile = "")
```

Description: Navigates to the specified view. If a video file is provided, plays it as a fullscreen transition before changing views. Right-clicking the video skips to the destination view immediately.

Parameters:

- `View` (viewnum): ID of the destination view
- `VidFile` (vidfile): Optional video file for transition effect

Restrictions: view, object, global, pocket

PlaySound

```
PlaySound(SoundFile: sndfile, Asynchronous: bool, Volume: 01float, Loop:
bool)
```

Description: Plays the specified audio file. Asynchronous playback returns control immediately; synchronous playback blocks until complete.

Parameters:

- `SoundFile` (sndfile): Path to the audio file
- `Asynchronous` (bool): If true, plays without blocking
- `Volume` (01float): Volume level (0.0 to 1.0)
- `Loop` (bool): If true, loops the audio continuously

Restrictions: view, object, global, pocket

PlayBackgroundMusic

```
PlayBackgroundMusic(SoundFile: sndfile, Volume: 01float, Loop: bool)
```

Description: Plays the specified audio file as background music. Only one background music stream can play at a time - calling this while music is playing stops the current music first. Background music persists across view changes and is not affected by StopAllSounds.

Parameters:

- `SoundFile` (sndfile): Path to the audio file
- `Volume` (01float): Volume level (0.0 to 1.0)
- `Loop` (bool): If true, loops the music indefinitely

Restrictions: view, object, global, pocket

StopBackgroundMusic

```
StopBackgroundMusic()
```

Description: Stops the currently playing background music.

Parameters: None

Restrictions: view, object, global, pocket

StopSound

```
StopSound(SoundFile: sndfile)
```

Description: Stops playback of the specified sound file if currently playing.

Parameters:

- `SoundFile` (sndfile): Path to the audio file to stop

Restrictions: view, object, global, pocket

StopAllSounds

```
StopAllSounds()
```

Description: Stops all currently playing sound effects. Does not affect background music.

Parameters: None

Restrictions: view, object, global, pocket

PlayVideo

```
PlayVideo(VidFile: vidfile, Start: number, Left: number, Top: number,
Volume: 01float, Loop: bool)
```

Description: Plays a video file at the specified position. Right-click to close the video.

Parameters:

- `VidFile` (vidfile): Path to the video file
- `Start` (number): Start time in seconds
- `Left` (number): X position
- `Top` (number): Y position
- `Volume` (01float): Volume level (0.0 to 1.0)
- `Loop` (bool): If true, loops the video

Restrictions: view, object, global, pocket

PlayVideoWithin

```
PlayVideoWithin(VidFile: vidfile, Start: number, WithinObject: objnum,
Volume: 01float, Loop: bool)
```

Description: Plays a video within the bounds of a specified object. The video is scaled to fit the object's dimensions.

Parameters:

- `VidFile` (vidfile): Path to the video file
- `Start` (number): Start time in seconds
- `WithinObject` (objnum): ID of the object to play within
- `Volume` (01float): Volume level (0.0 to 1.0)
- `Loop` (bool): If true, loops the video

Restrictions: view, object, global, pocket

StopVideo

```
StopVideo(VideoFile: vidfile)
```

Description: Stops playback of the specified video file.

Parameters:

- `VideoFile` (vidfile): Path to the video file to stop

Restrictions: view, object, global, pocket

StopAllVideos

```
StopAllVideos()
```

Description: Stops all currently playing videos.

Parameters: None

Restrictions: view, object, global, pocket

ShowImage

```
ShowImage(ImageFile: picfile, Left: number, Top: number, Duration:
float, Clickthrough: bool)
```

Description: Displays an image at the specified position for a given duration.

Parameters:

- `ImageFile` (picfile): Path to the image file
- `Left` (number): X position
- `Top` (number): Y position
- `Duration` (float): How long to display (seconds), 0 for permanent
- `Clickthrough` (bool): If true, clicks pass through the image

Restrictions: view, object

ShowImageWithin

```
ShowImageWithin(ImageFile: picfile, Left: number, Top: number, Duration: float, Clickthrough: bool, WithinObject: number, HideTarget: bool, Stretch: bool)
```

Description: Displays an image within the bounds of a specified object.

Parameters:

- `ImageFile` (picfile): Path to the image file
- `Left` (number): X offset within the object
- `Top` (number): Y offset within the object
- `Duration` (float): How long to display (seconds)
- `Clickthrough` (bool): If true, clicks pass through
- `WithinObject` (number): ID of the object to display within
- `HideTarget` (bool): If true, hides the target object
- `Stretch` (bool): If true, stretches image to fit object bounds

Restrictions: view, object

ShowObject

```
ShowObject(Object: objnum)
```

Description: Makes a hidden object visible.

Parameters:

- `Object` (objnum): ID of the object to show

Restrictions: view, object, global, pocket

HideObject

```
HideObject(Object: objnum)
```

Description: Hides a visible object.

Parameters:

- `Object` (objnum): ID of the object to hide

Restrictions: view, object, global, pocket

AllowTake

```
AllowTake(Object: objnum)
```

Description: Allows an object to be picked up and placed in a pocket.

Parameters:

- `Object` (objnum): ID of the object

Restrictions: view, object, global, pocket

DisallowTake

```
DisallowTake(Object: objnum)
```

Description: Prevents an object from being picked up.

Parameters:

- `Object` (objnum): ID of the object

Restrictions: view, object, global, pocket

SetVariable

```
SetVariable(Variable: varname, Value: value)
```

Description: Creates or updates a variable with the specified value.

Parameters:

- `Variable` (varname): Name of the variable
- `Value` (value): Value to assign

Restrictions: view, object, global, pocket

DelVariable

```
DelVariable(Variable: varname)
```

Description: Deletes the specified variable.

Parameters:

- `Variable` (varname): Name of the variable to delete

Restrictions: view, object, global, pocket

ClearKeyBuffer

```
ClearKeyBuffer()
```

Description: Clears all accumulated keystrokes from the key buffer.

Parameters: None

Restrictions: view, object, global, pocket

TextBox

```
TextBox(Message: value, Left: number, Top: number, Duration: float,
FontColor: fgcolor, BackColor: bgcolor, FontSize: fontsize, Bold: bool)
```

Description: Displays a text box with the specified message and styling.

Parameters:

- `Message` (value): Text to display
- `Left` (number): X position
- `Top` (number): Y position
- `Duration` (float): How long to display (seconds)
- `FontColor` (fgcolor): Text color
- `BackColor` (bgcolor): Background color
- `FontSize` (fontsize): Font size
- `Bold` (bool): If true, uses bold text

Restrictions: view, object, global, pocket

TextDialog

```
TextDialog(Message: value)
```

Description: Displays a modal dialog box with the specified message.

Parameters:

- `Message` (value): Text to display

Restrictions: view, object, global, pocket

InputDialog

```
InputDialog(Prompt: value, Variable: varname)
```

Description: Displays a dialog prompting the user for input. The entered value is stored in the specified variable.

Parameters:

- `Prompt` (value): Text prompt to display
- `Variable` (varname): Variable to store the input

Restrictions: view, object, global, pocket

SayText

```
SayText(Message: value)
```

Description: Uses text-to-speech to speak the specified message.

Parameters:

- `Message` (value): Text to speak

Restrictions: view, object, global, pocket

ShowURL

```
ShowURL(URL: value)
```

Description: Opens the specified URL in the default web browser.

Parameters:

- `URL` (value): URL to open

Restrictions: view, object, global, pocket

RunProgram

```
RunProgram(Application: exefile, Parameters: value)
```

Description: Launches an external application with the specified parameters.

Parameters:

- `Application` (exefile): Path to the executable
- `Parameters` (value): Command-line parameters

Restrictions: view, object, global, pocket

ChangeCursor

```
ChangeCursor(Cursor: cursor)
```

Description: Changes the mouse cursor to the specified style.

Parameters:

- `Cursor` (cursor): Cursor style identifier

Restrictions: (no restrictions)

HideMouse

```
HideMouse()
```

Description: Hides the mouse cursor.

Parameters: None

Restrictions: view, object, global, pocket

ShowMouse

```
ShowMouse()
```

Description: Shows the mouse cursor if hidden.

Parameters: None

Restrictions: view, object, global, pocket

HidePockets

```
HidePockets()
```

Description: Hides all pocket UI elements.

Parameters: None

Restrictions: view, object, global, pocket

ShowPockets

```
ShowPockets()
```

Description: Shows all pocket UI elements.

Parameters: None

Restrictions: view, object, global, pocket

Quit

```
Quit()
```

Description: Terminates the GEMS environment.

Parameters: None

Restrictions: view, object, global, pocket

Parameter Types

Type	Description
`viewnum`	View ID selector
`objnum`	Object ID selector

Type	**Description**
`value`	Text string
`varname`	Variable name
`number`	Integer value
`float`	Decimal number
`01float`	Float between 0.0 and 1.0
`bool`	Boolean (True/False)
`sndfile`	Audio file path
`vidfile`	Video file path
`picfile`	Image file path
`exefile`	Executable file path
`fgcolor`	Foreground color
`bgcolor`	Background color
`fontsize`	Font size
`cursor`	Cursor style
`key`	Keyboard key