Game Template Help

Setup Notes

Requirements: You must have the required plugins. If you do not have the Dark Data or Enhanced Animations plugin then comment out any functions that use Dark Data or Enhanced Animations.

- Matrix1Utils: http://forum.thegamecreators.com/?m=forum_view&t=85209&b=18
- Dark Data (Optional): http://www.thegamecreators.com/?m=view_product&id=2067
- Lua
  - Unity: http://www.thegamecreators.com/?m=view_product&id=2081
  - Barnski's Lua: http://forum.thegamecreators.com/?m=forum_view&t=74095&b=5

Debugger (Alpha): The debugger is both build into the template and the Command Center. The command center is for debugging, message logging, and error handing. You can also pause the game from the Command Center. The debugging features are currently in the alpha stage. This feature will be updated in future.

- **Error Reporting:** Reports errors and pauses the game. You can "[R]esume - [S]ave and Exit - [E]xit". "Resume" will ignore the error. "Save and Exit"will save the error log and shutdown the game. "Exit" will just shutdown the game.
- **Pause:** You can pause the game by pressing the P key on the keyboard when the Command Console window is in focus.
- **Console:** The warnings and messages can be outputted via console if ConsoleMode = 1.

Game Template Flags:

- **Display_FPS:** Displays the Frames Per Second in the right hand corner if set to 1.
- **DataMode:** A DataMode of 1 creates/loads a file database. A DataMode of 2 creates/loads a Dark Data database.
- **RebuildDatabase:** Removes the current database files so it can be rebuilt using the ScanFolders function.
- **LoadFileArray:** If you are loading by Database set this flag to one to also add the data to the File/Folder Arrays.
- **SetupAnimation:** This flag set to 1 will setup all the animation files automatically. Set to 0 to save on load time. This only needs to be run once then updated if the files change.
- **CharacterShop:** If you are using CharacterShop objects and have then setup correctly then set CharacterShop to 1.
- **ApocalypticPack:** If you are using ApocalypticPack objects and have then setup correctly then set ApocalypticPack to 1.
- **ModelPack10:** If you are using ModelPack10 objects and have then setup correctly then set ModelPack10 to 1.
- **DebugMode:** DebugMode will open the Command Center if set to 1. The errors, warnings, and messages will be sent to the Command Center if set to 1.
- **ConsoleMode:** If ConsoleMode is set to 1 then a console will be opened and display all important data including performance data.
- **WindowOn:** If WindowOn is set to 1 then the game will be displayed in a Window and if set to 0 it will be full screen.

Game Template Loop:

- This loop is run if the Escape Key is not pressed and the Game Template isn't shutting down
- In the loop it will update the performance code
- When the Escape Key is press or the Game Template is Shutting Down then it will run the close function

```
While EscapeKey() = 0 And ShuttingDown = 0
     Update_Performance()
EndWhile
```

Main Loop:

- This is the function that will be called by the performance function.
- The main loop is for logic

```
Function Loop_Main()
    Update_Main()
EndFunction
```

Display Loop:

- This is the function that will be called by the performance function.
- The display loop is for anything that needs to be rendered or outputted to the screen
- Put these commands in the Display Loop: Print, Sync, Camera, or Sprite Commands

```
Function Loop_Display()
    Update_Display()
EndFunction
```

Function Tags

Setup Functions:

- A function with the "Setup_" tag will be called first.
- The Setup functions are used to setup a class

Coroutines:

- A function with the "Coroutine_" tag is considered a couroutine
- A coroutine requires the Update_Coroutines() function to be called within the loop

Threads:

- A function with the "Thread_" tag is considered a Thread. (Used in Multi-Threading class)
- A Thread function requires this code:
    - `Sleep 250`

```
Index = Lua_GetThreadIndex("Thread_FunctionName")
```

```
Repeat
    If Get_CanRunFlag(Index) = 0 Then ThreadSleep(1.0)
    ` Do Stuff Here
    Set_CanRunFlag(Index, 0)
Until EscapeKey() = 1 Or ShuttingDown = 1
```

- You must sleep for 250 miliseconds or the Thread will crash
- You have to check to see if the game is Shutting Down and exit the thread when it does start shutting down
- If CanRun = 0 then it will Sleep so it can return the Time back to the processor
- After the code is run it has to reset the CanRun flag to 0

Loops:

- A function with the "Loop_" tag is considered a loop
- A loop is handled automatically by the performance function
- You can create more loops and set a custom Loops Per Second rate

Input:

- A function with a "KeyPressed_" tag is the Key Pressed event for when a key is pressed on the keyboard.
- A function with a "KeyReleased_" tag is the Key Released event for when a key is released on the keyboard.
- A function with a "KeyDown_" tag is the Key Down event for when a key is held down on the keyboard.
- A function with a "MousePressed_" tag is the Mouse Pressed event for when a Mouse Button is pressed on the mouse.
- A function with a "MouseReleased_" tag is the Mouse Released event for when a Mouse Button is released on the mouse.
- A function with a "MouseDown_" tag is the Mouse Down event for when a Mouse Button is held down on the mouse.
- A key is followed after the function tag for the keyboard events. (Uppercase)
- A button is followed after the function tag for the mouse events. (Left, Right, Middle)

Functions

Main.dba

- Loop_Main ()
    - This is the function that will be called by the performance function.
    - The main loop is for logic
- Loop_Display ()
    - This is the function that will be called by the performance function.

    - The display loop is for anything that needs to be rendered or outputted to the screen
    - Put these commands in the Display Loop: Print, Sync, Camera, or Sprite Commands
- Update_Main ()
    - Everything that pertains to the template that needs to be updated
- Update_Display ()
    - Everything that pertains to the template that needs to be updated or outputted

<u>**Animation.dba**</u>

- Setup_Animation ()
    - Extracts the animation and writes all animations exported to an animation list
- CharacterShop_Extract ()
    - Extracts the Character Shop animation and No Anim object file
- ApocalypticPack_Extract ()
    - Extracts the Apocalyptic Pack animation and No Anim object file
- ModelPack10_Extract ()
    - Extracts the Model Pack 10 animation and No Anim object file
- CharacterShop_WriteList ()
    - Exports the full animation file list in a text file
- ApocalypticPack_WriteList ()
    - Exports the full animation file list in a text file
- ModelPack10_WriteList ()
    - Exports the full animation file list in a text file
- CharacterShop_NoAnimAll ()
    - Exports a copy of the object with no animations
- ApocalypticPack_NoAnimAll ()
    - Exports a copy of the object with no animations
- ModelPack10_NoAnimAll ()
    - Exports a copy of the object with no animations
- Extract_FPSCreatorBase ()
    - Extracts the base animation of the FPS Creator animation list. The format is the same so I can extract the base and then the extra animations.
- Extract_CharacterShopBase ()
    - Extracts the base animation of the Character Shop animation list. The format is the same so I can extract the base and then the extra animations.

<u>**Dynamic Resources.dba**</u>

- Coroutine_GarbageCollection ()
    - This coroutine checks every index for empty indexes and adds them to the delete resource arrays

Delete Resource: Adds the index to the Deleted Resource Array and deletes/closes resource.

- DeleteObject (Index)
- DeleteSprite (Index)
- DeleteMemblock (Index)
- DeleteCamera (Index)
- DeleteEmitter (Index)
- DeleteVector2 (Index)
- DeleteVector3 (Index)
- DeleteVector4 (Index)
- DeleteImage (Index)
- DeleteEffect (Index)
- DeleteSound (Index)
- DeleteMesh (Index)
- DeleteMusic (Index)
- CloseFile (Index)
- DeleteMatrix (Index)
- CloseDFS (Index)
- CloseKFS (Index)
- DeleteLookup (Index)
- DeleteBank (Index)
- DeleteMutex (Index)

Next Resource: Gets the next resource index. If a deleted resource was added to the array then it will remove one from the stack and return that index making sure it fills empty indexes. **Returns** Index.

- NextObject ()
- NextImage ()
- NextCamera ()
- NextEffect ()
- NextSound ()
- NextMesh ()
- NextMemblock ()
- NextVector ()
- NextEmitter ()
- NextMusic ()
- NextFile ()
- NextSprite ()
- NextMatrix ()
- NextDatabase ()
- NextLookup ()
- NextBank ()
- NextCoroutine ()
- NextMutex ()

- NextDll ()

File Array Index: Assigns a filename to a File Array Index. FileGet returns the index from the filename given. Lua_FileGet(Name$) **Returns** Index

- Lua_FileAdd (Index, Name$)
- Index = Lua_FileGet (Name$)

File Load Resource: Loads the resource with that Name$ and **Returns** the Index

- File_LoadObject (Name$)
- File_LoadSound (Name$)
- File_LoadMusic (Name$)
- File_LoadEffect (Name$)
- File_LoadImage (Name$)
- File_LoadAnimation (Name$)

Lua Load Resource: Loads the resource with that Name$ and **Returns** the Index

- Lua_LoadObject (Name$)
- Lua_LoadSound (Name$)
- Lua_LoadMusic (Name$)
- Lua_LoadEffect (Name$)
- Lua_LoadImage (Name$)
- Lua_LoadAnimation (Name$)

Database Load Resource: Loads the resource with that Name$ and **Returns** the Index

- Database_LoadObject (Name$)
- Database_LoadSound (Name$)
- Database_LoadMusic (Name$)
- Database_LoadEffect (Name$)
- Database_LoadImage (Name$)
- Database_LoadAnimation (Name$)

Lua Get Resource Index: Finds a resource with the name Name$ and **Returns** the Index

- Lua_GetObject (Name$)
- Lua_GetImage (Name$)
- Lua_GetCamera (Name$)
- Lua_GetEffect (Name$)
- Lua_GetSound (Name$)
- Lua_GetMesh (Name$)
- Lua_GetMusic (Name$)
- Lua_GetFile (Name$)
- Lua_GetEmitter (Name$)
- Lua_GetSprite (Name$)

- Lua_GetMatrix (Name$)
- Lua_GetVector (Name$)
- Lua_GetDatabase (Name$)
- Lua_GetLookup (Name$)
- Lua_GetBank (Name$)
- Lua_GetCoroutine (Name$)
- Lua_GetMutex (Name$)
- Lua_GetAnimationByName (Name$)
- Lua_GetAnimation (Name$)

Lua Name Resource: Assigns a resource Index with a Name$ for easy lookup

- Lua_NameObject (Index, Name$)
- Lua_NameImage (Index, Name$)
- Lua_NameCamera (Index, Name$)
- Lua_NameEffect (Index, Name$)
- Lua_NameSound (Index, Name$)
- Lua_NameMesh (Index, Name$)
- Lua_NameMusic (Index, Name$)
- Lua_NameFile (Index, Name$)
- Lua_NameEmitter (Index, Name$)
- Lua_NameSprite (Index, Name))
- Lua_NameMatrix (Index, Name$)
- Lua_NameVector (Index, Name$)
- Lua_NameDatabase (Index, Name$)
- Lua_NameLookup (Index, Name$)
- Lua_NameMemblock (Index, Name$)
- Lua_NameBank (Index, Name$)
- Lua_NameCoroutine (Index, Name$)
- Lua_NameMutex (Index, Name$)
- Lua_NameAnimation (Index, Name$)

Resource Search: File does a search though the File() Array. Database does a search though the File Database. Lua does a search to find the File() Array Index. (Lua is quicker then File Search)

- Database_ResourceSearch (Name$)
- File_ResourceSearch (Name$)
- Lua_ResourceSearch (Name$)

Free Resource: Finds a empty Index by searching all Indexes. (Very slow)

- FreeObject ()
- FreeImage ()
- FreeCamera ()

- FreeEffect ()
- FreeSound ()
- FreeMesh ()
- FreeMusic ()
- FreeEmmitter ()
- FreeSprite ()
- FreeFile ()
- FreeMatrix ()
- FreeVector ()
- FreeLookup ()
- FreeMemblock ()
- FreeBank ()

Extra Functions: These functions are command shortcuts. They **Return** the Index

- LoadObject (Path$)
- LoadImage (Path$)
- LoadEffect (Path$)
- LoadSound (Path$)
- LoadMusic (Path$)
- LoadMesh (Path$)
- OpenFileToWrite (Path$)
- OpenFileToRead (Path$)
- CloseCurrentFile ()
- LoadSprite (Path$, X, Y)
- CreateSprite (X#, Y#, Image)
- CreateCamera ()
- CreateSphere (Size#)
- CreateComplexSphere (Size#, Rows, Columns)
- CreateCube (Size#)
- ObjectToMesh (Object)
- MeshToObject (Mesh)
- CreateAnimatedSprite (FileName$, Across, Down)
- LoadDll (DllName$)

Events.dba

Setup Events: Checks all the functions for tags and calls the appropriate functions to

- SetupEvents ()

Performance Events: Creates a performance event which is handled by the UpdatePerformance() function. SetThread function will set the performance event to a thread event. A thread event needs permission to run using the CanRun flag.

- PerformanceEvent_Create ()
- PerformanceEvent_SetThread ()

Event Functions: Creates, assigns parameters, and calls events. Everything to make events, assign parameters, and call them easier.

- Event_Create (FunctionName$)
- Event_Call (Index)
- Event_CallByName$ (FunctionName)
- Event_CallByPtr (Pointer)
- Event_CallByNumber ()
- Event_SetParameter1# (Number#)
- Event_SetParameter1$ (String$)
- Event_SetParameter2# (Number#)
- Event_SetParameter2$ (String$)
- Event_SetParameter3# (Number#)
- Event_SetParameter3$ (String$)
- Event_SetParameter4# (Number#)
- Event_SetParameter4$ (String$)
- Event_SetParameter5# (Number#)
- Event_SetParameter5$ (String$)

Coroutines: Creates a coroutine. Every coroutine must call Update_Coroutines() within the loop. This is automatically handled by the SetupEvents function by Tagging the function with the tag: "Coroutine_"

- CreateCoroutine (FunctionName$)
- Update_Coroutines ()
- Close_DeleteCoroutines ()

Threads (Alpha): Creates a new thread. This technique is called Multi-Threading. This is automatically handled by the SetupEvents function by Tagging the function with the tag: "Thread_"

- CreateThread (FunctionName$)
- CloseThread ()
- CloseAll_Threads ()
- ThreadSleep ()

Mutexes: Creates the mutex which is handled by the Threads. The CanRun flag controls the Threads performance which is handled by the UpdatePerformance() function. CreateMutex() is automatically called in SetupEvents() function.

- CreateMutex ()
- Set_CanRunFlag ()

- Get_CanRunFlag ()

Lua Name/Get Thread: Name a thread and **Returns** the Index of a thread by giving it a Name$. Lua_NameThread () is automatically called in CreateThread() function when a new thread is created.

- Lua_GetThreadIndex (Name$)
- Lua_NameThread (Index, Name$)

<u>File.dba</u>

- Setup_File ()
    - If the database doesn't exists then it will scan. If it exist then it will load the database.
- SaveAsDatabase ()
    - Saves the Database using Dark Data
- SaveAsFile ()
    - Saves the Database using a File
- LoadFile ()
    - Loads the File Database
- LoadDatabase ()
    - Loads the Dark Data Database
- File_WriteList (FolderName$, FolderPath$)
    - Writes a list of every file in that folder
- ScanFolder (Dir$)
    - Scans every folder and every folder inside that folder for files and folders and writes them to the corresponding arrays
- HandleFile (SubLevel)
    - Determines if it is a file or folder. Adds to corresponding array.
- RemoveExtension$ (FilePath$)
    - Removes the extension from the path or file name.
- GetFileExtension$ (FilePath$)
    - Gets the file extension from the path or file name.
- GetPathFolder (Level, Path$)
    - Gets a folder from a path. The level is the folder Index starting from the end of the path.
- GetExtentionType (Extention$)
    - Gets the extension type and returns the extension name.
- DeleteFileDatabase()
    - Deletes the database

<u>Input.dba</u>

- Thread_ScanInput ()
    - Will scan the keyboard and mouse for input can call the appropriate functions/events.
    - This is a thread meaning it will automatically be created and run parallel with the main thread.

- KeyPressed (Code)
  - Will check the Keys() array for any keys that match the Code given. Will **Return** the state of 1 if the Key is pressed.
- MousePressed (Code)
  - Will check the Mouse() array for any buttons that match the Code given. Will **Return** the state of 1 if the Button is pressed.
- Mouse_GetState (Code)
  - Checks the current pressed mouse button and compares it with the code given. This will work if multiple keys are pressed returning 1 if the button is pressed.
- MouseName$ (Code)
  - **Returns** the mouse Name$ from the Code given
- FindKeyIndex ()
  - Finds an empty Keys() index
- FindMouseIndex ()
  - Finds an empty Mouse() index

Messages.dba: Sends messages to the control center and also receives messages and the messages can control the game in some way.

- Send_Error$ (Message$)
  - Sends an error to the command center. Will pause the game and ask if the game should be continued.
- Send_Message$ (Message$)
  - Sends an message to the command center
- Send_Warning$ (Message$)
  - Sends an Warning to the command center
- Handle_Messages ()
  - Handles all incoming messages
- Send_ShutDown ()
  - Sends the shutdown flag so the command center shuts down

Performance.dba: Performance controls the rate at which each loop runs and also the rate of the threads.

- Update_Performance ()
  - Does two things: Controls the rate in Loops Per Second and returns the time nothing is running back to the processor. This ensures a low CPU usage.

Setup.dba: Sets up every class and the game.

- Setup ()
  - Master Setup() function. Call before the main loop.
- Close ()
  - Master Close() function. Call after main loop.
- Lua_LoadScript ()
  - Loads the master Lua Script.

Timers.dba: For creating Game Timers and Timers with Events.

- Create_Timer ()
  - Creates a new Timer.
- Create_TimedEvent ()
  - Creates a new Event Timer.
- Update_TimedEvents (EventIndex, StartTime#, Units$, ResetFlag)
  - When the countdown timer reaches 0 it will call the event and restart if the ResetFlag is set to 1
  - The EventIndex is the Index of an Event() to call
  - The Units$ are: “Hours”, “Minutes”, “Seconds”, “MiliSeconds”
  - StartTime# is the time will count down from
  - ResetFlag if set to 1 then the countdown timer will restart
- Reset_Timer (Index)
  - It will restart the timer back to 0
- GetHours (Index)
  - Gets the time in hours
- GetMinutes (Index)
  - Gets the time in Minutes
- GetSeconds (Index)
  - Gets the time in Seconds
- GetMilliSeconds (Index)
  - Gets the time in Milliseconds
- GetFreq ()
  - Gets the processor frequency