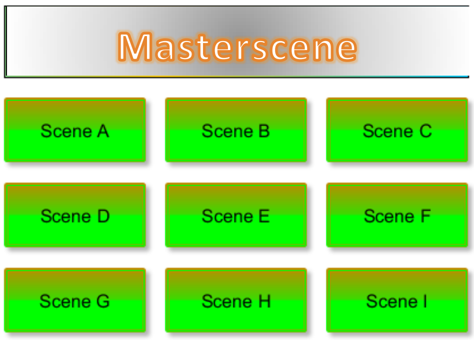
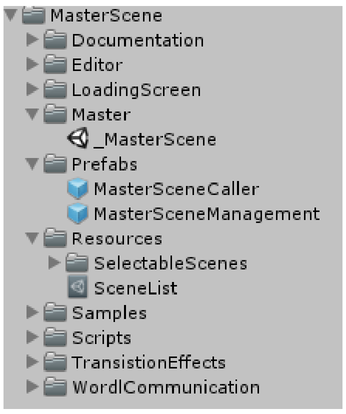
Masterscene

Introduction:

I am developing games and applications in unity for about 2 Years now and I must admit that Unity is the best Game Engine available except when it comes to one thing: The scene system. The main Problem is that I want to have objects like UI which contain cheat options or player information in every scene. The first option is to make Singletons for every class but it gets messy very fast. After a time I came up with the solution by implementing a Master Scene and it solved all my problems.



What does it:

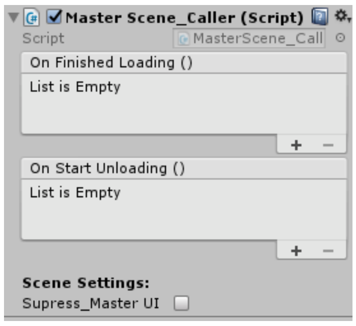


```
▼ MasterScene
  ▶ Documentation
  ▶ Editor
  ▶ LoadingScreen
  ▼ Master
      _MasterScene
  ▼ Prefabs
      MasterSceneCaller
      MasterSceneManagement
  ▼ Resources
    ▶ SelectableScenes
      SceneList
  ▶ Samples
  ▶ Scripts
  ▶ TransistionEffects
  ▶ WordlCommunication
```

The Masterscene simply is a scene above all other scenes. This means, that this scene is available forever, no matter what`s happening. It will be loaded when the game starts and unloaded when you close the application. The result is that every object located in the masterscene is active the whole game (like a singleton). Objects which should be in the masterscene are all kind of persistent stuff like save systems, projectile systems or score counters. An additional feature is the easy implementation of transition effects and loading bars which are also implemented in the masterscene. This Asset contains 3 transition effects which are described below as a sample.

How to use:

Setup normal scenes and Masterscene_Caller:



Master Scene_Caller (Script)

Script MasterScene_Call

On Finished Loading ()

List is Empty

On Start Unloading ()

List is Empty

Scene Settings:

Supress_Master UI

In Order to use this system, every normal scene must contain one MasterScene_Calle r object. So drag one MasterScene_Caller prefab into the scene if it doesn't contain one.

This object is responsible for adding the MasterScene if it is the first scene loaded. It also contains optional Scene Settings which may have more options in the future. Additionally it contains 2 Unity events which can be used by the event system which are:

- On Finished Loading: Is called, when the MasterScene Manager has loaded a new Level.
- On Start Unloading; Is called before the scene gets unloaded.

Note: If an attached function throws an exception, the error message is generated which tells you that the bad function is assigned to one of those events. It tells you which event caused the error and in which scene:

```
"One or more functions assigned to the Event: OnStartLoading/ OnFinishedLoading
(MasterScene_Caller) has thrown an exception in the [CURRENTSCENE]"
```

The Scene Settings currently only contains the Option to suppress the UI in the master scene and is a way to add your custom options. This option is there as it is needed to disable persistent User Interfaces in some scenes like Splash Screens or Title Screen.

The most important functions of the MasterScene_Caller are Reload() and ChangeScene(string). Reload simply reloads the current scene. It is useful in cases where you just want to reload the scene like when the player dies etc.

The ChangeScene(string) function does what its name says and It invokes a scene change where the string is the Name of the scene. When changing scene, a verification check is done to prove that the scene change is authentic.

The ChangeSceneNoLoadingEffect(string) function does what its name says and It invokes a scene change but without executing Loading Effects. This is useful when the loading times are so short that you don't have to display loading gauges.

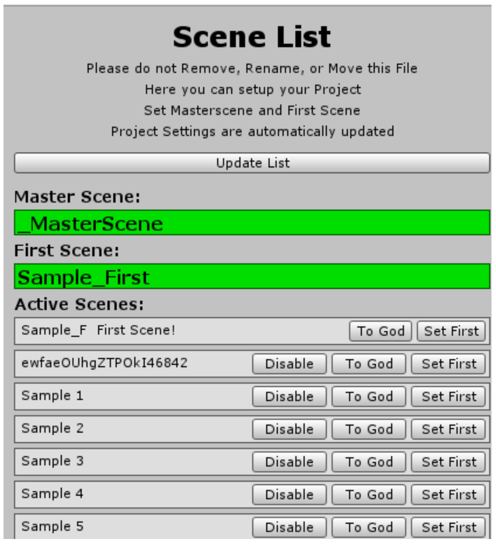
Possible Validation fails are:

- SceneList does not exist: If you managed to delete or rename the SceneList file somehow, create one by going to your Project tab and select Create > Custom > MasterScene > SceneList. Make sure it is in a resource folder.

- Scene is not in the SceneList: Is your string Correct?
- Scene is in List but does not exist: Scene got removed or renamed?
- Another Scene is already loading: Safety measure for people who bash buttons -,- wait until the loading is finished ☺

Scene List:

The most important object for this system is the SceneList file which is located in the Resource folder. Please do not DELETE IT or RENAME IT and it has to be in a Resource folder! I even point it out as some testers really removed this file. If you still removed it somehow, you can create a new one by selecting:


Scene List

Please do not Remove, Rename, or Move this File
Here you can setup your Project
Set Masterscene and First Scene
Project Settings are automatically updated

Update List

Master Scene:
_MasterScene
First Scene:
Sample_First
Active Scenes:

Sample_F First Scene!		To God	Set First
ewfaeOUhgZTPOkI46842	Disable	To God	Set First
Sample 1	Disable	To God	Set First
Sample 2	Disable	To God	Set First
Sample 3	Disable	To God	Set First
Sample 4	Disable	To God	Set First
Sample 5	Disable	To God	Set First

Create – Custom – MasterScene – SceneList in the Project Tab where you create Materials etc.

If you have a new scene or renamed a scene, the SceneList has to be updated. This is done manually by pressing the “Update List” Button or automatically when you hit the play button as the MasterScene_Manager will update the SceneList when you start your game in the Unity Editor. You actually only have to check it when you want to set the first scene which is usually a splash screen or title screen. It will also add the scenes in the build settings so you don’t have to do it manually when building the game (I would always forget it and end up with missing scenes in the build).

Every entry can be disabled, enabled or Set as First or Master Scene by clicking on the buttons.

When updating, the system will scan the complete Asset Database for scene files and adds them into the list. All scenes get an entry in the Active Scenes section unless they are marked as disabled. For example, there are 3 Scenes which should be hidden in the image right.

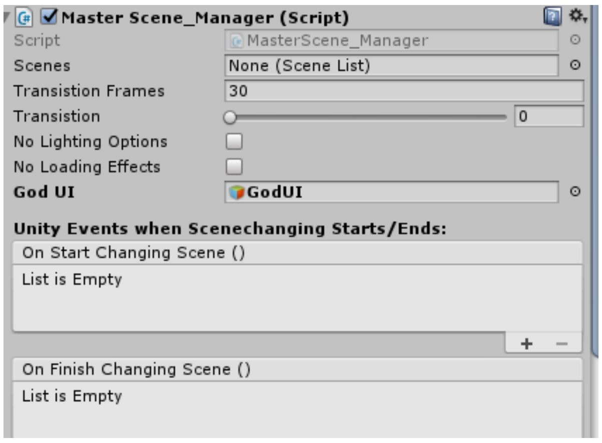
There is an entry for the Master Scene. This tells the name of the MasterScene. Either keep it and use the master scene from the sample or create your own.

Another entry is the First Scene. This describes which scene should be loaded at first when building the game. This scene will be the first entry in the scene list located in the build settings.

Now there is also the option to turn off the auto updating when hitting the play button as it slows down starting the game in editor when the project is huge.

Master Scene Management:

This object has to be located in the Master Scene. It manages all scene changings and implements transition effects. It is implemented as cheap singleton _Instance to just be able to access it from other scenes. You want to access it when implementing stuff which should change scenes. When you want to implement your own stuff like loading bars and transition effects, several events with their according UnityEvents are exposed which are:



Master Scene_Manager (Script)

Script	MasterScene_Manager
Scenes	None (Scene List)
Transistion Frames	30
Transistion	0
No Lighting Options	☐
No Loading Effects	☐
God UI	GodUI

Unity Events when Scenechanging Starts/Ends:

On Start Changing Scene ()
List is Empty

On Finish Changing Scene ()
List is Empty

- `_onFinishedLoadingLevel`: Is called when a loading process is finished and the scene is loaded. It is called after everything is loaded to make sure every object is really called. Is Called after Start() so everything is there
  - According UnityEvent: OnStartChangingScene

- `_onBeginUnloadingLevel`: Is called before the scene transition starts. It is called before OnDestroy(). Use this to important scene related stuff like saving. Everything is still alive when this is called
  - According UnityEvent: OnFinishChangingScene

Note: If an attached function throws an exception, the error message is generated which tells you that the bad function is assigned to one of those events:

"One or more functions assigned to the Event: OnStartChangingScene/ OnFinishChangingScene (MasterScene_Manager) has thrown an exception in the Masterscene!"

Events for Transitioning:

- `_onStartTransistioning`: Is called when Transistioning sets in. Use this to enable your transitioneffect GameObject
  - According UnityEvent: OnStartTransition

- `_onTransistioning`: Is called while changing scene. Has a value which is between 0 and 1. 0 means no loading is going or it is finished. 1 means scene is loading.
  - According UnityEvent: OnTransition

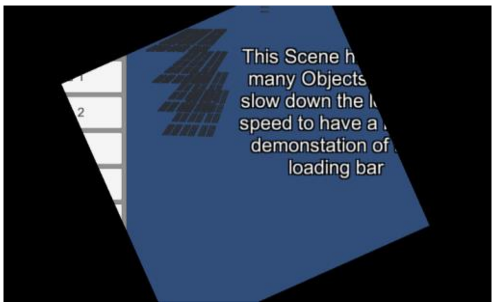
- `_onFinishTransistioning`: Is called when Transistioning sets is finished. Use this to disable your transitioneffect GameObject
  - According UnityEvent: OnFinishTransition

Events for Loading Effects/Screens:

- `_onStartLoading`: Is called when Loading Effect sets in. Use this to enable your LoadingBar/Screen GameObject
  - According UnityEvent: `OnStartLoadingScene`

- `_onLoading`: Is called while scene is loading. Has percent between 0 and 1 where 0 means 0% loaded and 1 means 100% loaded. Is only useful if scenes have long loading times. The sample scenes are loaded way to fast in order to see the example loading bar in action.
  - According UnityEvent: `OnLoadingScene`

- `_onFinishLoading`: Is called when Loading Effect sets in. Use this to disable your LoadingBar/Screen
  - According UnityEvent: `OnFinishLoadingScene`



The Transition Frames describe how fast the transition effect should be. This is in Frames which means that 60 Frames = 1 Second if Fixed Time step is 60 Frames per second. It currently contains a Master Ui as parameter which is for User Interface which must be persistent through all scenes. The MasterScene_Caller will deactivate this game object when the flag SupressMasterUI is true.

The important functions are ChangeScene(String) and Reload which do the same thing described in the section Setup normal scenes and MasterScene_caller. It also has the ChangeSceneNoLoadingEffect(String) version.

Notice: When you load a scene, the lighting option of the loaded scene is used unless you set the Flag “`NoLightingOptions`”. If flagged, the Lighting Options of the Master Scene will be used

If you set the “NoLoadingEffect” Flag, no loading effect will occur even if changescene is called.

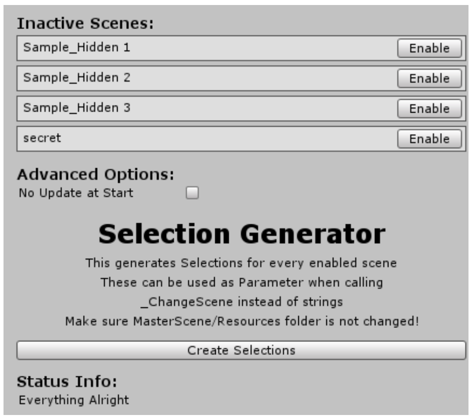
The function AddScene(string) and SubtractScenes(string) should only be used by experienced uses. It simply adds or removes a scene. Use these functions if your game has to load multiple scenes. However this is highly experimental as I never needed such loading system. These Functions don’t do scene transitions and event calling. It only verify that the scene you want to load exists.

The last function is a simple SimulateTransistioning(). It simply does the effect without changing any scene.

Selection Generator:

Using strings as input parameter always is a bad thing as simple typos are enough to create errors and humans tend to make mistakes.

Therefore is the possibility to create Scene selectors. These are files which only contain the name of a scene and can be used as input parameter instead of strings. These files are created in the "/Assets/MasterScene/Resources" folder and can be created by hitting the "Create Selections" button in the scenelist.


Inactive Scenes:

Sample_Hidden 1	Enable
Sample_Hidden 2	Enable
Sample_Hidden 3	Enable
secret	Enable

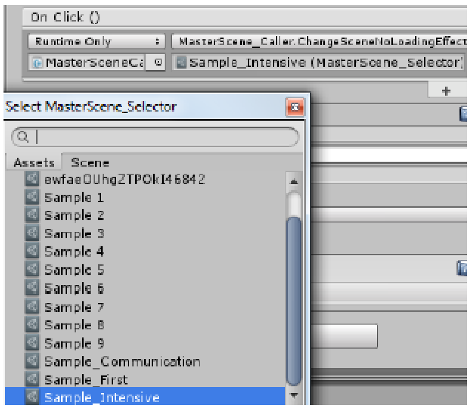
Advanced Options:
No Update at Start ☐

Selection Generator
This generates Selections for every enabled scene
These can be used as Parameter when calling
_ChangeScene instead of strings
Make sure MasterScene/Resources folder is not changed!

Create Selections

Status Info:
Everything Alright

To use these objects, simply call the _Changescene(MasterSceneSelector) and select your scene in the asset database like in this picture. You can see a scene with a very ugly name. Imagine the struggle when you have type this as string.


On Click ()
Runtime Only | MasterScene_Caller.ChangeSceneNoLoadingEffect
MasterSceneCa | Sample_Intensive (MasterScene_Selector)

Select MasterScene_Selector
Assets | Scene
- ewfaeOUhgZTPOkI46842
- Sample 1
- Sample 2
- Sample 3
- Sample 4
- Sample 5
- Sample 6
- Sample 7
- Sample 8
- Sample 9
- Sample_Communication
- Sample_First
- Sample_Intensive

Keep in mind that these are not changed when you rename or delete a scene file. They are just like strings and if the scene does not exist, you get the standard message that the scene does not exist. Also when you remove a Selection from the folder, the input parameter of your Portal/Button is null. This throws this exception:

ArgumentException:Failed to convert Parameters ………

This exception is thrown somewhere in the UnityEvent system so I cannot change it to an useful error message like "You want to change scene with selectors but you don't have one assigned"

Tipps for Extending:

It is easy to extend this system by adding your custom transition effects and loading screen. To do this, use the Events provided by the MasterScene_Manager. If you have to access objects which are in the master scene, make one game object in the master scene and add a simple script to it which has a static reference to it and your stuff assigned as member variable to it.

When you want to create your own transition effects, you may investigate how stencil system works in unity. This sample contains 2 Materials which represent this to create this rotating square transition.

Setup your own master scene:

The easiest way to setup your own master scene is just to copy the existing one and update the scenelist It's not worth your time to create a new scene, throw in MasterScene_Manager prefab, setup UI stuff when you just can copy the sample master Scene it.

Possible Errors/Dangers:

My scene does not load:

- Check if your scene has a MasterScene_Caller if not, add it to the Scene by simply dragging the prefab into the scene.
- Check if the MasterScene_Management is in the master scene.
- Check if SceneList exists. If not create it (Create > Custom > MasterScene > SceneList in project tab) and move it in a resource folder.
- Check if the Scene you want to load exists.

UI does not work:

- This happens if you make a canvas where Unity automatically creates an EventSystem object. Delete it if it already exists in the master scene.
- Make sure there is no Label in the Master Scene which blocks ray casts over the whole screen.
- Loading Bar not visible: Make sure it does not get obstructed by transition effects.

Unity Freezes (Substance):

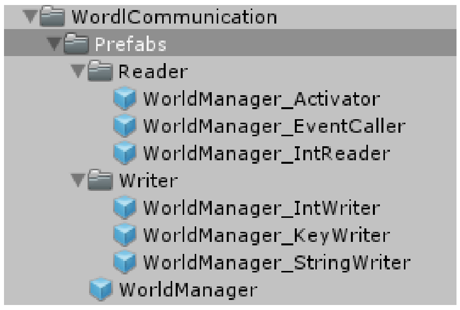
- Before 2018.1 Substance files (.sbsar) were supported by unity. Unity sometimes froze completely when a scene was loaded asynchrony which contained substance materials. Support for substance got removed in 2018.1 and must be obtained from the asset store. The solution is to convert generated textures to normal textures which can then be used.

Multiple Audio Listeners:

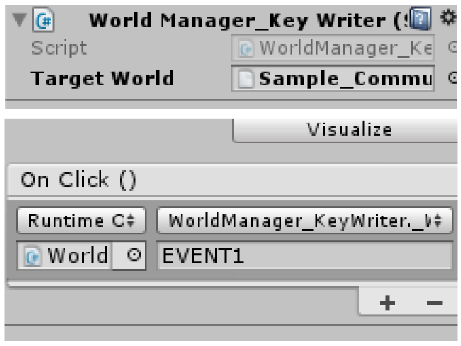
- A camera in Unity has automatically an Audio Listener. Make sure there are no multiple Audio Listeners. Either delete the Listener in the master scene or remove it from cameras in your scenes.

World Manager:

This system shows how easy it is to extend the current system with an own plugin which allows communication between scenes. In order to set it up, create the MasterScene_Selections by using the Selection Generator described in previous page. Then simply throw in the WorldManager prefab into the master scene. The world manager has no settings and its inspector will only show written entries during play mode. Keep in mind that this system currently doesn't stay persistent and world information is lost when you exit the play mode. Save Systems may be implemented in further updates as it can also take advantage of the master scene system but there already exists a numerous amount of save systems in the asset store



```
▼ WordlCommunication
  ▼ Prefabs
    ▼ Reader
        WorldManager_Activator
        WorldManager_EventCaller
        WorldManager_IntReader
    ▼ Writer
        WorldManager_IntWriter
        WorldManager_KeyWriter
        WorldManager_StringWriter
      WorldManager
```



World Manager_Key Writer (
Script: WorldManager_Ke
Target World: Sample_Commu

Visualize

On Click ()
Runtime C | WorldManager_KeyWriter._V
World | EVENT1

The database has a category for each scene you have. In order to write information into the database, throw in one of those three sample prefabs into your current scene. Every writer needs a MasterScene_Selector as target world which is required or it will not write anything. In order to write something into the database, simply call "*_WriteKey(string key)*" from another script or by using the Event System like in the right image. Keep in mind that writer like the IntWriter has to set the Value which should be written into the database.

Entries can also be removed from the database by simply calling _RemoveWorldInformation(string key) which can also be called by using EventSystems.

Writer Classes

All writer inherit from the WorldManager_BaseWriter class. The base class simply implements the *_RemoveWorldInformation*, a *_Write(key, Value)* function and *_UpdateWorld* function. Every derived class should use the _Write function in order to write Information into the database. Following writer are currently implemented:

- **KeyWriter:**

This class simply implements a _Write(string Key) function which simply writes the key into the database where the Key is also the Value. Useful

- **StringWriter**

Similar to KeyWriter but has a Value property with an added function _SetValueToWrite(string Value). The _Write(string Key) method writes the Value with Key into the database.

- **IntWriter**

This writer writes an Interger into the database and is similar as the StringWriter.

Extending:

Every .net class and Serializable class can be written into the database. However keep in mind that such entries also require a reader which can work with those informations.

Readers:

Reader will read the information which is stored in the database. Whenever a scene is fully loaded, the WorldManager fires the „_onApplyWorldInformation event which also includes the Information about the current loaded world. The sample readers currently implemented are following entries:

- **Activator:**

This basic reader simply checks if the key exists and enables or disables assigned game objects. If the "Inverted" flag is set, the opposite is true and objects are enabled if the key doesn't exist

- **Eventcaller:**

This reader behaves similar to the Activator and just checks if the key exists or not. It has 3 UnityEvents which are "OnExisting"," OnNotExisting" and "DynamicState(bool)" and are fired according to result. If "removekeyafterwards" is set, the key is removed if exists.

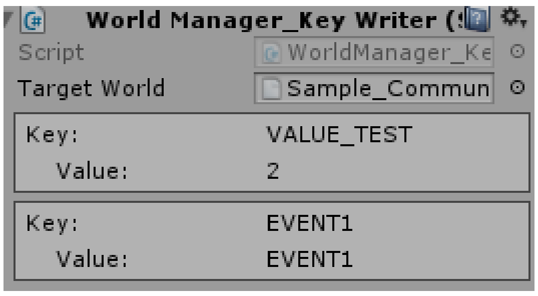
- **IntReader:**

As any object can be written into the database, different reader are required to use them. This reader reads the value of an entry and fires an Existing_Int and Existing_float if the data type is really "int" and nothing else.

Extending:

Any class can implement the `_onApplyWorldInformation(Dictionary<string, object> Information)` event.

It is important that your entry could consist of any class. Therefore it is recommended to use a Try-Catch block in order to prevent InvalidCastException errors like it is implemented in IntReader. It can happen that an entry has the wrong data type when it first get written by an IntWriter and then an StringWriter overwrites the entry with an "string" data type if both writer have set the same key as parameter.



World Manager_Key Writer (

Script	WorldManager_Ke
Target World	Sample_Commun
Key:	VALUE_TEST
Value:	2
Key:	EVENT1
Value:	EVENT1

The written entries are visible in the inspector during play mode. Writer only show entries of their targeted world and the WorldManager itself shows all existing entries.

Last Notes:

If you have any questions, suggestions, bug reporting don't hesitate to contact me. If you are going to sell a game which uses this asset, inform me because I may buy your game and play it ☺

Contact Information:

E-Mail: mhartl.mmt-b2013@fh-salzburg.ac.at

Homepage: http://lostinpixels.org/