Bs

# kBinaryBitmap Saver 1.0

UNITY3D EXTENSION

**kBinaryBitmapSaver** V1.0
**Manual** 1.0
**Online Documentation** klock.de/klock/sources

**Paul Knab** - Apr 2012

# Table of Contents

Introducing 1

Import into your project 1

What comes with 1

How to open the Editor Window 1

The Main Window - Overview 2

The Main Window - Workflow 3

The Export Extensions Window - Overview 5

The Export Extensions Window - Workflow 6

The Texture Importer Settings Window 7

The Binary Bitmap Viewer 8

The using of kBinaryData 9

Using an resource Holder object 10

API Class Reference - kBinaryData 12

## Introducing

**kBinaryBitmapSaver** is a tool for creating byte array files from bitmaps. The tool copies a selected bitmap file, from the user environment, into a folder in your unity project. It will be converted into a byte array and will be stored with a selectable extension, in a data file.

The process of the texture import can adjusted in the *Texture Importer* window. The extension of the export data array file can be customized within the *Export Extension* window.

It is recommended to work with bitmap files in the following formats ( bmp, gif, jpg, png, psd, tga, tiff, iff, pict ).

### Import into your unity project

#### Import Custom Package

1. *Unity Toolbar* ***Windows*** **>** ***Assets*** **>** ***Import Package*** **>** ***Custom Package***

2. Browse to the ***kBinaryBitmapSaver.package*** File and import the data.
   The package will import the required assets into their correct locations.

#### Import Source Files

1. Right mouse click in the ***Project View***.

2. Browse to your unity project .. */Assets/[foldername].*

3. Extract the ***kBinaryBitmapSaver folder*** from the
   ***BinaryBitmapSaver-src.rar*** file, in the .. */Assets/[directoryname]*

### What comes with

When the import of the package is complete the project view should look like this.

- *Assets/*
  - ▼ *kBinaryBitmapSaver/*
    - *Gui*
    - *kBinaryBitmapExtension*
    - *kBinaryBitmapImporter*
    - *kBinaryBitmapSaver*
    - *kBinaryBitmapViewer*
    - *kBinaryBitmapData*

### How to open the kBinaryBitmapSaver Editor

1. *Unity Toolbar* ***Klock*** **>** ***kBinaryBitmapSaver*** **>** ***Converter***
2. Then the **kBinaryBitmapSaver** editor opens.

## The Main window - Overview

The editor consists 3 views.

### Target Folder

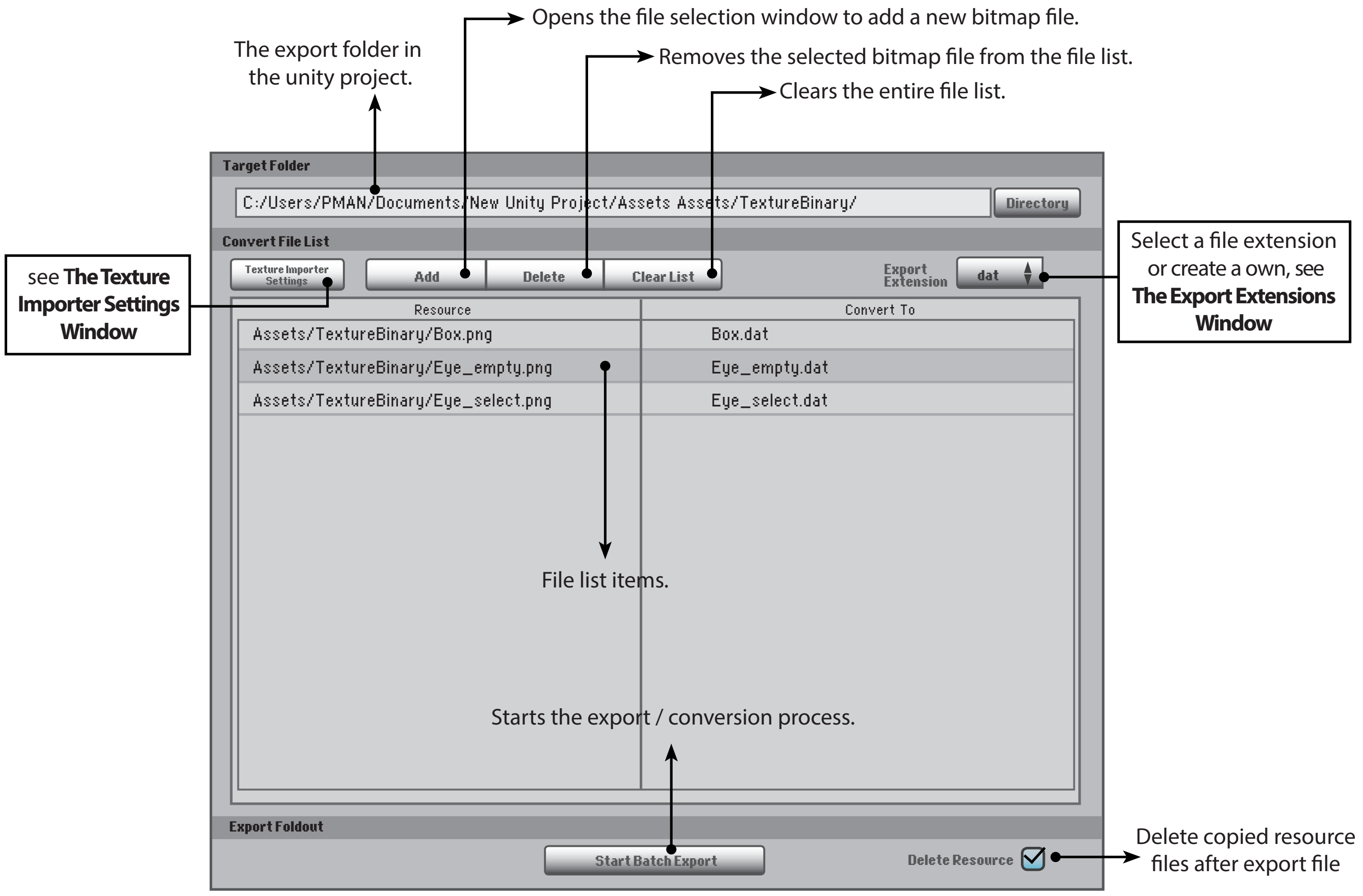
Shows the insertion path for the bitmap files to be copied.
Points to the export path for the converted data files.
With the *Directory* button, the path of *Target Folder* can be changed.

### Convert File List

Draws a list of bitmap files to be converted, in a binary data file.
Shows a menu to add and remove bitmap files.
Denotes the export extension slot, where you can define the file extension.
( see **The Export Extensions Window** )
Includes a button to open the *Texture Importer Settings* window.
( see **The Texture Importer Settings Window** )

### Export View

The view contains a button that launches the batch conversion process.
With a toggle, one can specify whether the bitmap resource files will be deleted after the batch process. ( for copies of external bitmap files only )



Opens the file selection window to add a new bitmap file.
The export folder in the unity project.
Removes the selected bitmap file from the file list.
Clears the entire file list.
Target Folder
Directory
Convert File List
Texture Importer Settings
Add
Delete
Clear List
Export Extension
dat
see **The Texture Importer Settings Window**
Select a file extension or create a own, see **The Export Extensions Window**
Resource
Convert To
File list items.
Starts the export / conversion process.
Export Foldout
Start Batch Export
Delete Resource
Delete copied resource files after export file

C:/Users/PMAN/Documents/New Unity Project/Assets Assets/TextureBinary/

| Resource | Convert To |
|---|---|
| Assets/TextureBinary/Box.png | Box.dat |
| Assets/TextureBinary/Eye_empty.png | Eye_empty.dat |
| Assets/TextureBinary/Eye_select.png | Eye_select.dat |

## The Main Window - Workflow

### Select a new Target Folder

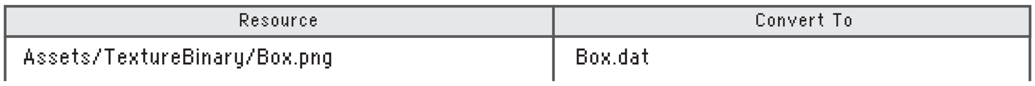
1. Click the Directory button and the folder selection dialog opens.

2. Select a folder or create one, as a new work path.

### Adding a bitmap file

1. Click the Add button and the file selection dialog opens.

2. Select a bitmap, in the format of ( bmp, gif, jpg, png, psd, tga, tiff, iff, pict ) and press *Open*. **kBinaryBitmapSaver** stores a copy of the bitmap file into the *Target Folder.*



| Resource | Convert To |
|---|---|
| Assets/TextureBinary/Box.png | Box.dat |

3. The file list now displays the new resource item.
   The first column shows the copied bitmap file.
   The final export file, with its extention is shown in columns two.

**Note:** If you have copied resource data from the unity project, the bitmap file will not be saved in the *Target Folder*, only the converted binary data file.

### Remove a bitmap file

1. Select the row you want to remove.

2. Click the Delete button and a dialog appears.

3. You can choose between two options :

| | |
|---|---|
| Item and list | Delete the item in file list and the file from the *Target Folder*. |
| Item | Delete the item only from the file list. |

### Clear the complete list

1. Click the Clear List button.

2. See the steps of **Remove a bitmap file**. But the change affects the whole file list.

## The Main Window - Workflow

### Select an export extension

1. Click on the Export Extension dat popup and a list appears.

2. Select a list item with the wanted export extension.

### Adjust the Texture Import Settings

1. Click the Texture Importer Settings button and the settings window opens.

2. Set your desired settings.

3. Click on Apply to finalize your settings and return to the main window.

### Automatically removing of the resource files

**kBinaryBitmapSaver** can automatically delete the resource files, which is no longer needed after the conversion. When the Delete Resource option is disabled, no bitmap files are deleted after the batch process.

1. Activate the Delete Resource ☑ toggle.

### How to convert bitmap files

To convert a bitmap file into byte array data file.

1. Add the bitmap files you want to convert.

2. Setup the *Texture Importer Settings*.

3. Choose your desired *Export Extension*.

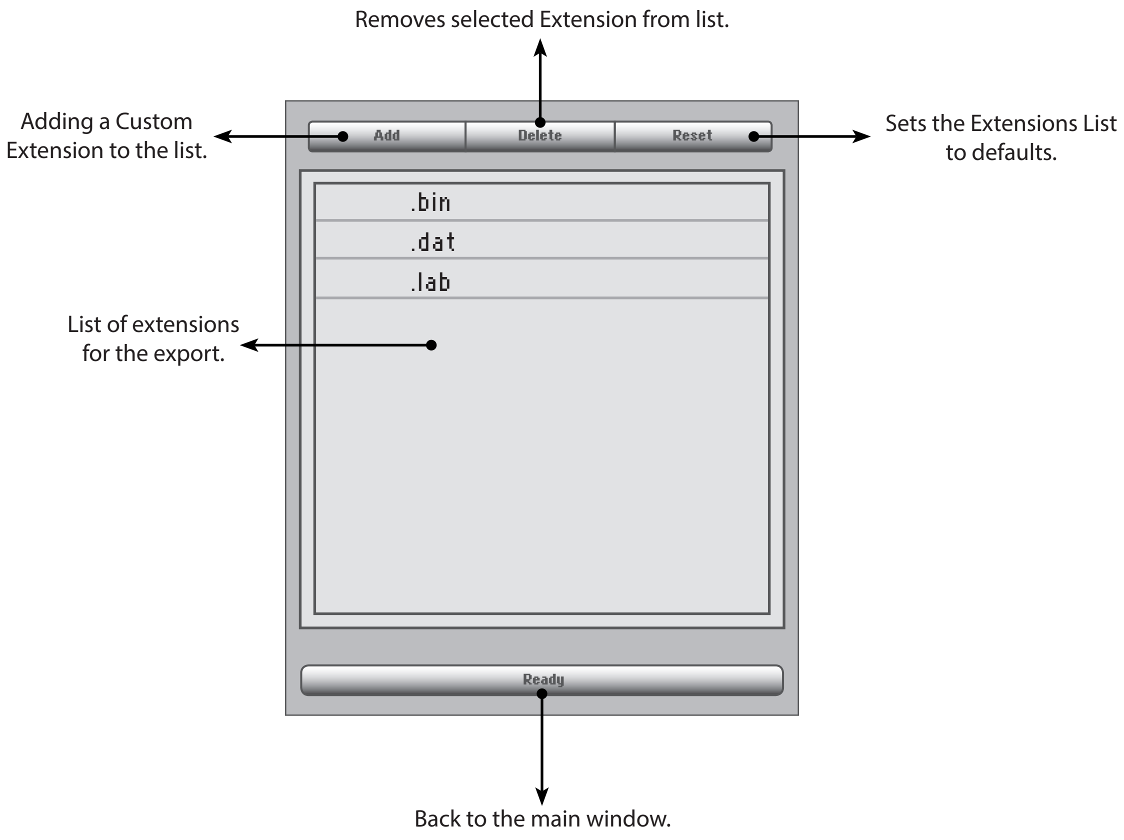
4. Select if the Delete Resource option is wanted.

5. Click Start Batch Export to begin the batch process.

## The Export Extensions Window

### Overview

To create a custom format, use the *Export Extension window*. The window shows the usable extensions.  Additionally, you can create own extensions, remove extensions from the file list, or if anything goes wrong, reset the extension list to the editor defaults formats.


Removes selected Extension from list.
Adding a Custom Extension to the list.
Add
Delete
Reset
Sets the Extensions List to defaults.
.bin
.dat
.lab
List of extensions for the export.
Ready
Back to the main window.

## The Export Extensions Window

### Workflow

#### Opening the Export Extensions Window

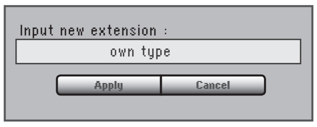
1. Press the Export Extension dat pop up in the *main window* and a list will appear.

2. Select the create own item and the *extension window* opens.

#### Adding a Custom Extension

1. Press Add in the *extensions window* to create a new file extension.

2. Type a valid name string and finish with a press on **Apply**.


Input new extension :
own type

3. The extensions list now displays the new export file extension.

#### Remove an extension

1. Select the extension, that you want to delete.

2. Click the Delete button, in the *extension window*, to remove a format from the list.

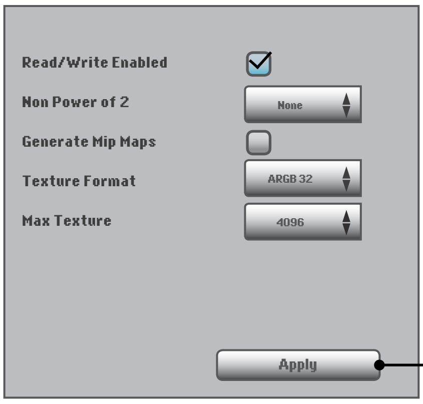
#### Reset to Default Extensions

1. In order to reset to the default editor extentions, press the Reset button in the *export extension window*.

## The Texture Importer Settings Window

### Overview

With the *Texture Importer Settings* window, you can quickly adjust your own values  for the import of textures into *Unity3D*. For the using of *Texture Import Settings* window see **Adjust the Texture Import Settings**.


Read/Write Enabled ✓
Non Power of 2: None
Generate Mip Maps
Texture Format: ARGB 32
Max Texture: 4096

Use the selected parameters and return to the main window.

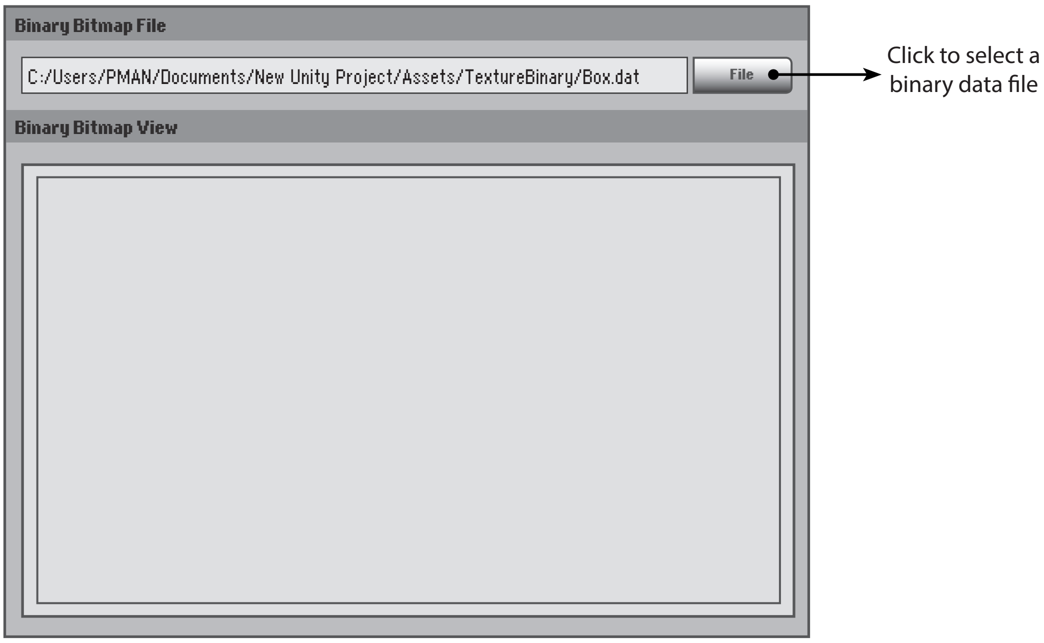
| | |
|---|---|
| **Read/Write Enabled** | Select this option in order to enable access to the texture data from scripts. |
| **Non Power of 2** | When texture has non-power-of-two size, this option will define a scaling behavior at import time.<br><br>**None** Padded to the next larger power-of-two size (for *GUITexture*)<br>**To nearest** Scaled to the nearest power-of-two size at import time<br>**To larger** Scaled to the next larger power-of-two size at import time<br>**To smaller** Scaled to the next smaller power-of-two size at import time |
| **Generate Mip Maps** | Select this to enable mip-map generation. Mip maps are smaller versions of the texture that are used when the texture is very small on screen. |
| **Texture Format** | Defines what internal representation is used for the texture.<br>This is a tradeoff between size and quality.<br><br>Selectable formats:<br><br>**ARGB32** Truecolor with alpha. 256 KB for a 256x256 texture.<br>**RGB24** Truecolor but without alpha. 192 KB for a 256x256 texture. |
| **Max Texture** | Setting the maximum imported texture size. |

## The Binary Bitmap Viewer

### Overview

If you have more than one data file, it may be advantageous to have a file viewer. The **kBinaryBitmapSaver** package includes a very simple file viewer. The **kBinaryBitmapViewer** shows a binary bitmap data file as a *Texture2D* in a resizable view. It also shows information about the *Texture2D* object.

Note: Only works with the following file extensions ( .dat, .bin, .lab ).


Binary Bitmap File
C:/Users/PMAN/Documents/New Unity Project/Assets/TextureBinary/Box.dat
File
Click to select a binary data file
Binary Bitmap View

### How to open the kBinaryBitmapViewer

1. *Unity Toolbar **Klock** > **kBinaryBitmapSaver** > **Viewer***
2. Then the **kBinaryBitmapViewer** opens.

### View a binary bitmap file

1. Click the File button and then select a binary bitmap data file in the appearing file selection panel.

2. The viewer converts the binary data to a *Texture2D* and displays the result in the bitmap viewer.

## The using of kBinaryData

To use a binary file in a script, have a look at the *kBinaryData.cs*. It has features to handle binary data.( see **API Class Reference** )

In this example, we use the *Loader* () function to convert a texture from a binary bitmap data file.

1. Create a C# script and name it "*SimpleLoader.cs*".

```
using UnityEngine;
using klock;

public class SimpleLoad : MonoBehaviour
{
    private Texture2D texture = null;

    void Start ()
    {
        if( texture == null )
        {
          texture = kBinaryData.Loader( Application.dataPath + “/TextureBinary/Box.dat“ );
        }
    }

    void OnGUI ()
    {
        if( texture != null )
        {
          GUI.DrawTexture( new Rect( 50, 50, texture.width, texture.height ), texture);
        }
    }
}
```

2. Create a new *GameObject* and reset the position.

3. Name the new *GameObject* “Looker“.

4. Drag the *SimpleLoad.cs* from the ***Project View*** to the *GameObject* “Looker“ in the ***Inspector Panel***.

5. Press the ***Play*** button in the unity Toolbar and change to the ***Game View.*** If it works correctly, the texture will be shown as a *GUITexture*.

## Using an resource holder object

Now we want to use a dictionary that serves as the central location to get resources from. This holder object would pool the assets and makes sure that textures are reused, rather than reloading them.

Access this class with the function *GetTexture* (*string* filename). The function checks whether the dictionary already has an entry to our texture filename or not. If it is the case the value of *Texture2D* is returned, otherwise a new *Texture2D* entry will be deserialized in the *kBinaryData.Loader* function and stored in the Dictionary holder object.
In addition, the class has for each data file a *public static string*, which refers to the data file path. So we can simply call the path of a file by *SampleAssets._Box*, instead of specifying the full path *Application.dataPath + foldername / file name.dat*.
The *public static string globalPath* points to the folder, that contains the binary data files to load.

1. Create a C# script with the name "*kAssets.cs*".

```
using UnityEngine;
using System.Collections;
using klock;

/** Class to hold content in a dictionary for efficient load and unload objects.*/

public class SampleAssets
{
     // folder to the binary data files
     public static string globalPath = Application.dataPath + "/TextureBinary/";

     // path to each data file - for easy handling
     public static string _BOX   = globalPath + "Box.dat";
     public static string _EYE_0 = globalPath + "Eye_empty.dat";
     public static string _EYE_1 = globalPath + "Eye_select.dat";

     // holder object for the textures
     private static IDictionary sTextures = new Hashtable();

     public static Texture2D GetTexture( string name )
     {
        if ( sTextures[name] == null )
        {
           sTextures[name] = kBinaryData.Loader( name );
        }
        return sTextures[name] as Texture2D;
     }
}
```

## Using an resource Holder object

2. Create a C# script with the name "*AssetsLooker.cs*".

```
using UnityEngine;
using klock;

public class SampleAssetsLooker : MonoBehaviour
{

     private bool showIt = false;

     void Start ()
     {
     }

     void OnGUI ()
     {
        GUI.DrawTexture( new Rect( 50, 50, 125, 125 ),
                         SampleAssets.GetTexture( SampleAssets._BOX ));

        if( GUI.Button( new Rect( 350, 50, 125, 125 ), „“, GUIStyle.none ))
        {
            showIt = !showIt;
        }

        GUI.DrawTexture( new Rect( 350, 50, 125, 125 ), SampleAssets.GetTexture(
                                ( showIt ) ? SampleAssets._EYE_1 : SampleAssets._EYE_0 ));
     }
}
```

3. Create a new *GameObject* and reset the position.

4. Name the new *GameObject* "AssetLooker".

5. Drag the *AssetsLooker.cs* script from the ***Project View*** to the *GameObject* “AssetLooker“ in the ***Inspector Panel***.

6. Press the ***Play*** button in the unity Toolbar and change to the ***Game View.***

# API Class reference - kBinaryData.cs

Deserialize a byte array, from a file, to Texture2D or serialize a Texture2D to byte array and store it in a file

## Static Public **Member Functions**

| | |
|---|---|
| static **void** | **Save** ( Texture2D picture, string filename ) |
| static **Texture2D** | **Loader** ( string filename ) |
| static **string** | **CutFile** ( string file ) |

## Static Public **Attributes**

| | |
|---|---|
| static **string** | **fPath** = "/TextureBinary/" |

## Static Private **Member Functions**

| | |
|---|---|
| static void | **Serialize** ( string filename, object objectToSerialize ) |
| static T | **Deserialize< T >** ( string filename ) |

## **Member Function** Documentation

### static void **Save** ( Texture2D picture, string filename )

Save a Unity Texture2D to a given filename and path.

Parameters : picture is Textur2D
filename is String

See also: **Serialize**( filename, objectToSerialize )

### static void **Serialize** ( string filename, object objectToSerialize )

Serialize a Unity Texture2D to a byte array and write it into a file.

Parameters : filename is String
objectToSerialize is object

### static Texture2D **Loader** ( string filename )

Load a data file and deserialized it to a Unity Texture2D.

Parameters : filename is String

Returns: The converts Bitmap Texture2D.

### static T **Deserialize< T >** ( string filename )

Open a byte array data file and deserialized it to a Unity Texture2D.

Parameters : filename is String

Returns: The converts Texture Serialize Object.

### static string **CutFile** ( string file )

Cuts a complete environment path conform to the Application.dataPath.
C:/Space/UnityProjects/DemoProject/Assets/myFolder -> Assets/myFolder

Parameters : filename is String

Returns: The path in your Unity project Assets/myFolder