PDF Renderer

Support
dave.paroxe@gmail.com

Table of Contents

1. PDFRenderer API

1.1 PDFDocument

This class represents a pdf document. It allows to render a page in a new 2DTexture, in an already existing 2DTexture, and to get information on the documents and pages.

1.1.1 Constructors

PDFDocument(byte[] buffer, string password)
PDFDocument(string filePath, string password)

Pdf document class constructors. The pasword field can be null even if the document is not protected.

1.1.2 Properties

bool IsValid

This property indicate if the document is in fact valid. It can be invalid when the buffer or the path is invalid. It can also be invalidif the document is protected of itf the specified password is invalid.

1.1.3 Methods

int GetPageCount()

Get the page number in the document.

int GetPageWidth(int pageIndex)
int GetPageHeight(int pageIndex)
Vector2 GetPageSize(int pageIndex)

These methods allow to have the size of the page specified in parameter.

Texture2D RenderPageToTexture(int pageIndex, RenderSettings renderSettings)
Texture2D RenderPageToTexture(int pageIndex, int width, int height, RenderSettings renderSettings)

Creates the page render in a new texture according to the rendering options specified in parameter. Do not forget to release this texture when not any more needed. Here is how to release the texture which does not serve any more

```
DestroyImmediate(tex);
Resources.UnloadAsset(tex);
```

void RenderPageToExistingTexture(int pageIndex, Texture2D texture, RenderSettings renderSettings)

Allows to render in an already existing texture. For a faster render, the texture should be in the RGBA32 format and does not contain MipMap. Should the opposite occur, PDFRenderer do the render using another method supporting more formats but a lot slower. Here is the instanciation of an ideal texture for PDFRenderer:

```
Texture2D tex = new Texture2D(width, height, TextureFormat.RGBA32, false);
```

1.1.4 Example of use

Here is an example of use of PDFDocument :

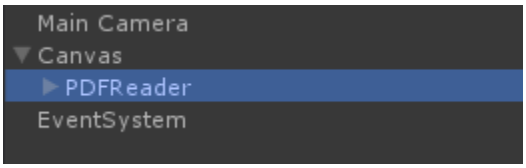
C#

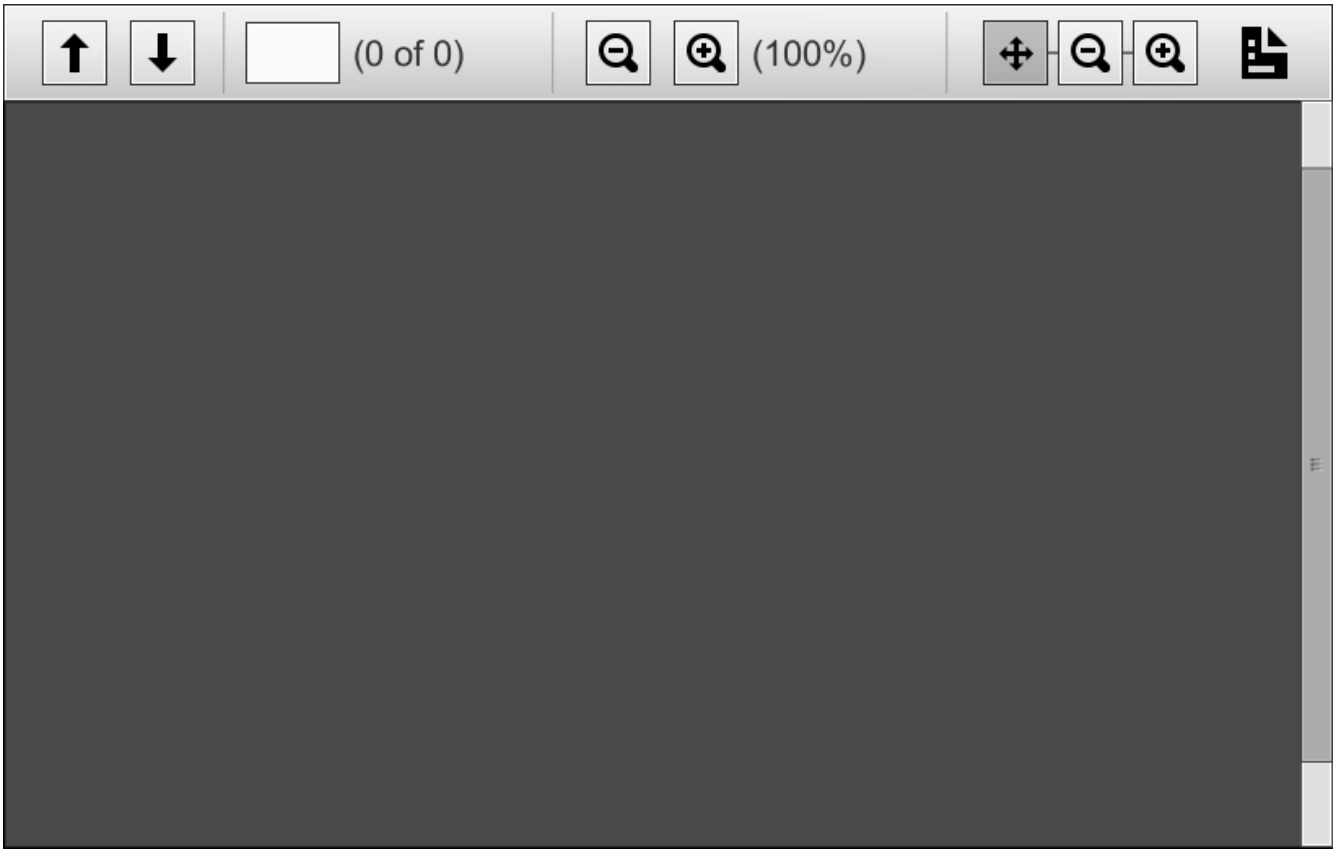
```
PDFDocument pdfDocument = new PDFDocument("pdf_sample.pdf", "");

Texture2D tex = pdfDocument.Renderer.RenderPageToTexture(3, 1024,

1024); GetComponent<MeshRenderer>().material.mainTexture = tex;
```

Figure 1: Example of use of a pdf applied on 3D geometry.

2. PDFViewer

PDFViewer is a control working with the new Unity (4.6+) UI system . A prefab is included in the package. You only need to drop it into the scene under a canvas, which is located in : Paroxe/PDFRenderer/Prefabs.





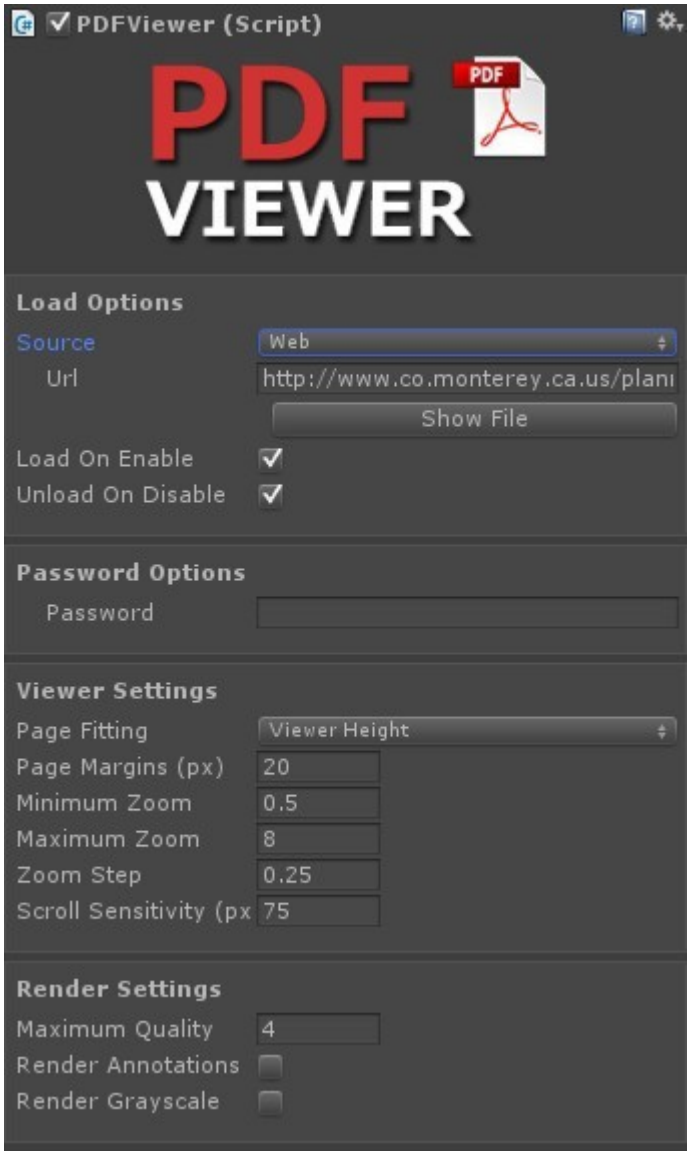
PDFViewer is easy to modify. It supports to be anchored, like all the control of the new UI system.

You can even arrange it in the form of a « World Space » by configuring the canvas in the following way :

This allows to use PDFViewer in a computer in a game as an example :

2.1 PDFViewer inspector

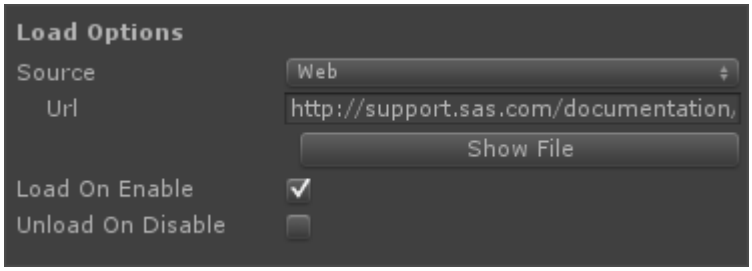
Here is the PDFViewer inspector. It was created in order to simplify its use as much as possible.



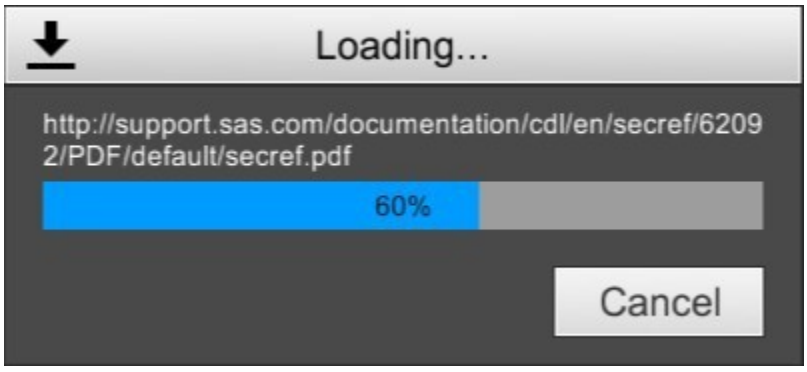
2.1.1 Loading options

PDFViewer offers several loading options in order to specify the source from where to load the pdf file to be shown.

2.1.1.1 Loading from the web

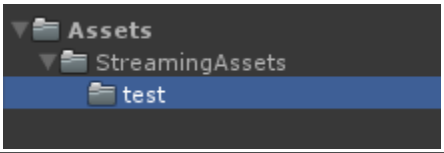


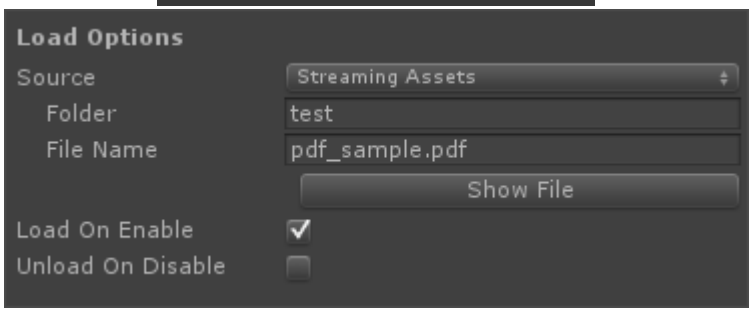
With this loading mode, PDFViewer load the pdf file from the web and informs the user of the downloading progress.



2.1.1.2 Loading from the StreamingAssets folder

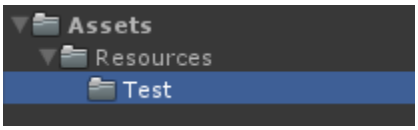
With this loading mode, PDFViewer loads the file from the StreamingAssets folder, which is a folder that must be created in Assets.

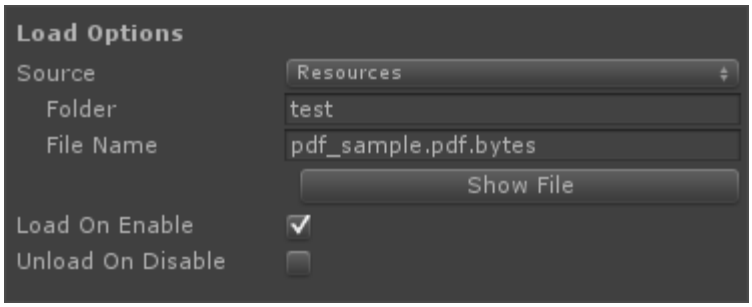




2.1.1.3 Loading from the Resources folder

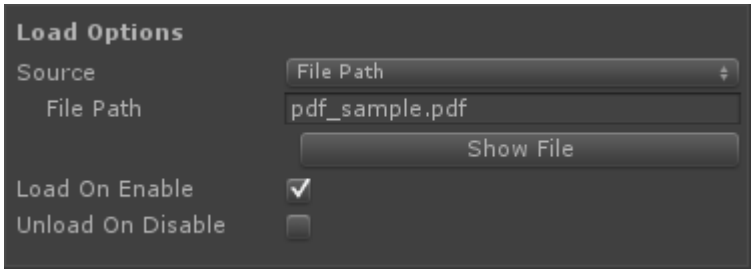
With this loading mode, PDFViewer loads the file from the StreamingAssets folder, which is a folder that must be created in Assets. Your pdf file should have « .bytes » as the extension in order to be read.





2.1.1.4 Loading from the path of a file

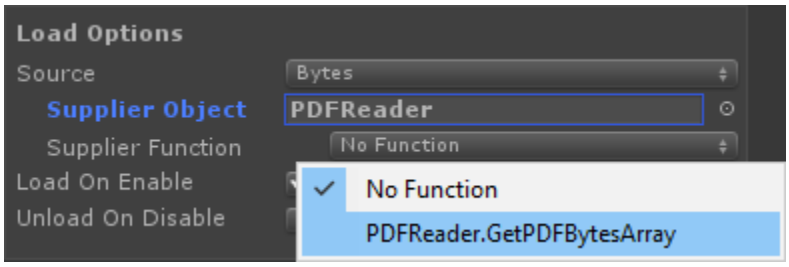
With this loading mode, PDFViewer load the file from a relative or absolute path provided in parameters. When a relative path is provided, it is relative to the project file, that is beside the Assets file. The « Show File » button allows to open the explorer and to select the file. It assures you that the informations seized are exact.



2.1.1.5 Loading from a BytesArray

This mode is very practical if none of the other modes meets your needs. You only have to specify a GameObject that implement a method with no arguments returning a BytesArray.

```
public byte[] GetPDFBytesArray()
```



2.1.2 Viewer settings

Viewing options allows to change the way the document is displayed while also allowing to change the way the user browses inside of it.

Page Fitting : Allows to configure how the pages adjusts according to the viewer (Viewer Width, Viewer Height, Whole Page, Zoom). The choice of the Zoom allows to specify the initial zooming factor.

Page Margins : Pixel margin between the pages.

Minimum/Maximum Zoom : Allowed zooming factor.

Zoom Step : Zooming interval at each zoom-in and zoom-out. Note that this factor is doubled when the zoom is at least at 200% and is quadrupled when the zoom is equal or higher than 400%.

Scoll Sensitivity : Specify the amount of scrolling when using the mouse wheel.

2.1.3 Rendering options

Maximum Quality : When the user zooms in the document, PDFViewer redo all the pages with a higher zooming factor so that the whole becomes more visible. However, if the user zooms a lot, the zooming factor can be so high that the textures becomes too bug, therefore slow to generate. That is why this option allows to specify a maximum to this quality. As an example, if this parameter is set to 4, the quality will not increase whe the user will zoom beyond an enlargment factor of 4.

Figure 3: Maximum Quality: 4, Zoom: 800%

Figure 2: Maximum Quality: 8, Zoom: 800%

Render Annotations : Allows the render of annotations included in the pdf file (highlighting, underlining etc.)

Render Grayscale : Allows to render the pdf in grayscale.