ULTIMATE
Screenshot Creator

v1.4.3

for Unity 4.6 to 2017.3

January 9, 2018

Contents

Thank you for purchasing *Ultimate Screenshot Creator*! I wish this package will meet your needs and expectations. Please do not hesitate to contact me if you have a feature request, or any question, issue or suggestion.

Arnaud,
support@wildmagegames.com

1 OVERVIEW

1.1 Introduction

Ultimate Screenshot Creator is the most complete and customizable screenshot creator available on the asset store. It is the ideal tool to create professional marketing and PR assets, wallpapers, mobile store screenshots, and more. It is also ideal to allow your users to take screenshots using your application, on all plaftorms. It also automatically adds your screenshots on iOS and Android galleries.

1.2 Features

- Perfectly works with Unity 4.6 and later UI system.
- Capture multiple resolution screenshots in one click.
- (NEW)Export your screenshots to Android Gallery.
- (NEW)Export your screenshots to iOS Camera Roll.
- (NEW)Capture sreenshots in-game on all platforms.
- Customizable set of cameras with custom rendering properties, such as culling mask, clear mode, clear color and field of view.
- Customizable overlay system to automatically include your game logo, watermarks, and more.
- In-game preview with photography guides to better frame your screenshots.
- Burst mode not to miss the best moments, or to be used as input of a GIF creator software.
- Ultra HD screenshots.
- (NEW)Capture only a sub-part of the screen.
- Powerful naming system with symbols to customize the export folder and file names.
- Easily export screenshots for all Stores (Amazon, Google Play, App Store, Windows Store, etc.)
- Use the delegates to call your code during the capture process.
- (NEW)Display a validation UI to the user to save the screenshot or discard it.
- Export to PNG and JPG.
- (NEW)Export the screenshots to platforms Picture Folder.
- Robust transparent backgrounds solution.
- Landscape and portrait mode.
- Screenshot preview, with photography guides.
- Resolution scaling.

- Customizable shot sound.
- Customizable hotkeys.
- Customizable export folder.
- Align cameras to view utils.
- Time management utils.
- Increment file name or override existing files.
- (NEW)Compatible with the new Unity post process stack.
- (NEW)Support multi-display settings.
- (NEW)Support of WebGL.

1.3 Requirements

- The solution works with Unity 4.6 and later free or pro edition.
- The capture mode *RENDER_TO_TEXTURE* requires Unity Pro, or Unity 5.0 and later free edition.
- On Android, to export to the mobile phone picture folder, your application needs to have external storage access rights. You can set it on the Player Preferences.
- On iOS, adding the picture to the phone gallery requiers the Photo Gallery access rights. This is done automatically by the script *iOsPostProcessBuild.cs*. If you intend to build for iOS but do not want to have this requirement, simply remove the *iOsPostProcessBuild.cs* script and the *iOSUtils.m* iOS plugin.
- We advise to use Unity 5.4 and later for a better GameView management using *GAMEVIEW_RESIZING* in editor (see FAQ 5.7).

1.4 Limitations

- Unity WebPlayer platform is not supported.

2 QUICK START GUIDE

2.1 Quick Configuration

1. Put the *ScreenshotManager* prefab into your scene.
2. Select the *ScreenshotManager* to configure it.
3. Select the capture mode. *GAMEVIEW_RESIZING* is the most robust capture mode, and captures the UI perfectly. It works in Editor and Windows Standalone only. *RENDER_TO_TEXTURE* works in editor and on all platforms, but Screenspace Overlay UI are not rendered. *FIXED_GAMEVIEW* works in editor and on all platforms, can only capture at the current screen resolution. It is the recommended settings for taking screenshots in-game.
4. Set the destination folder and the filename.
5. Select the camera mode: *GAME_VIEW* mode copies what you see in game view, *CUSTOM_CAMERAS* mode allows you to use a set of cameras. In custom mode, select the cameras to be used for the capture.

   Note that you can also customize their rendering settings.
6. Select the resolutions mode: *GAME_VIEW* mode uses the game view resolution, *CUSTOM_RESOLUTIONS* mode allows you to customize the resolutions to be used. In custom mode, select the resolutions to be used for the capture, or use the list + to create a custom resolution.
7. The overlay system enables the easy integration of your company or game logo into each screenshot. Edit the example canvas or create your own to display your game logo. Select the overlays to be used for the capture. You can customize the default overlay or create a new one using a *Canvas*.
8. The photography guide helps framing the screenshots, and is particularly useful using *PreviewWhilePlaying*.
9. That's it. Now just click on the *Take screenshot(s)* button, or use the capture hotkey (F12 by default, only works while playing).
10. Do not forget to Apply your modifications to the prefab to save them.

2.2 Platform Specific Configuration

This section details the platform specific configuration required to be able to take screenshots in-game.

2.2.1 Android

1. In the Player Preferences, set Write Access to External.
2. The capture mode the capture mode $FIXED\_GAMEVIEW$ is recommended to capture exactly what is on screen. $RENDER\_TO\_TEXTURE$ is possible too.
3. The export mode must be $PICTURE\_FOLDER$ or $PERSISTENT\_DATA\_PATH$.

The permission to access the image gallery should be asked at the first app launch.

Note that $PICTURE\_FOLDER$ tries to export to the primary storage, if available. That means that screenshots saved to that directory are not deleted if the app is uninstalled.

If the primary storage is not available, or on $PERSISTENT\_DATA\_PATH$, the plugin save to the first media storage available. On that case, the pictures are visible on the gallery, but can be deleted from the device when the app is uninstalled.

2.2.2 iOS

1. Move the *Plugins* folder at the root of the *Assets* folder. The plugin should be located exactly here: *Assets/Plugins/iOS/iOSUtils.m*.
2. On Unity 5.6 and later, check that the iOSUtils.m plugin has the dependency to the **Photos** framework (in rarely used frameworks).
3. The capture mode the capture mode $FIXED\_GAMEVIEW$ is recommended to capture exactly what is on screen. $RENDER\_TO\_TEXTURE$ is possible too.
4. The export mode must be $PICTURE\_FOLDER$ or $PERSISTENT\_DATA\_PATH$.

The permission to access the camera roll should be asked at the first screenshot capture. To ask the permission at startup, add the script *RequestAuthAtStartup* to your project, or call *iOsUtils.RequestGalleryAuthorization()* when you want to.

2.2.3 WebGL

1. Move the *Plugins* folder at the root of the *Assets* folder. The plugin should be located exactly here: *Assets/Plugins/WebGL/WebGLUtils.jslib*.
2. The capture mode the capture mode $FIXED\_GAMEVIEW$ is recommended to capture exactly what is on screen. $RENDER\_TO\_TEXTURE$ is possible too.
3. Note that with WebGL, the export mode is ignored since the image will be downloaded using the web browser.

3 CONFIGURATION GUIDE

3.1 Capture Mode

Figure 1: Capture mode settings.

There are two capture modes:

- *GAMEVIEW_RESIZING* is the most robust capture mode, and captures the UI perfectly. It works in Editor and Windows Standalone only. Note that with this mode the editor gameview or game windows needs to be resized for a few milliseconds.
- *RENDER_TO_TEXTURE* works in editor and on all platforms, but Screenspace Overlay UI are not rendered.
- *FIXED_GAMEVIEW* works in editor and on all platforms, can only capture at the current screen resolution. It is the recommended mode for taking screenshot in-game on all platforms.

	GAMEVIEW_RESIZING	*RENDER_TO_TEXTURE*	*FIXED_GAMEVIEW*
Custom Cameras	√	√	√
Custom Resolutions	√	√	Screen resolution
UI	√	No UI	√
Overlays	√	No overlays	√
In Editor	√	√	√
In Game	Windows Standalone only	All platforms	All platforms

Table 1: Capture modes comparison.

3.2 Destination

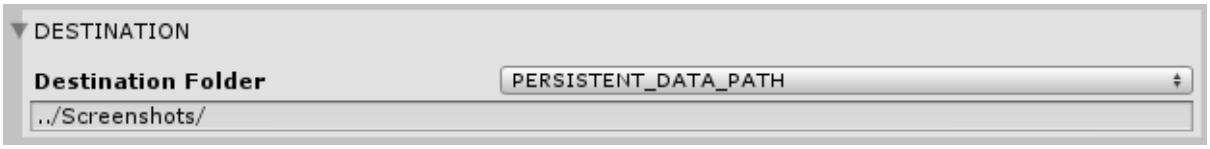

Figure 2: Destination settings.

There are three export modes:

- *CUSTOM_FOLDER* allows you to select the export folder.
- *DATA_PATH* exports the screenshots relatively to the project data path.
- *PERSISTENT_DATA_PATH* exports the screenshots relatively to the persistent data path.
- *PICTURE_FOLDER* exports the screenshots relatively to the platform picture folder. It is the recommended mode for taking screenshot in-game on all platforms.

3.3 File Name

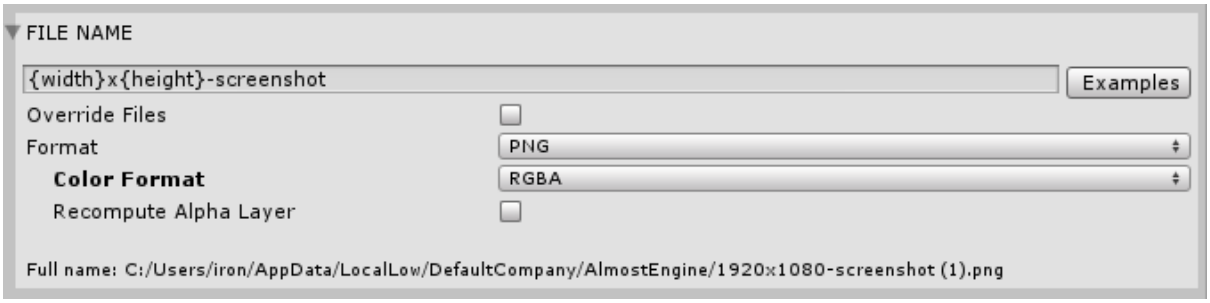

Figure 3: File name settings.

FileName. Defines the file name to be used for the screenshots. Use the *Examples* button to select one of the name presets.

You can use and combine the following symbols to better match your needs:

- {year}, {month}, {day}, {hour}, {minute}, {second} for the current time information,
- {width}, {height}, {scale}, {ratio}, {orientation}, {ppi}, {percent}, {name}, {category} for the resolution information.

Note that you can use the file name to group screenshots by resolution folders, etc. For instance {*width*}-{*height*}*/screenshot.png* will create one folder by resolution and create one *screenshot.png* in each of them.

Override Existing Files. By default, if you try to create a screenshot file that already exists, its name will be incremented. For instance: screenshot.png, screenshot(1).png, ... Set *override* to *true* if you want to override the existing files.

Format. You can export to *PNG* or to *JPG* with a custom quality.

Color Format. In *PNG*, you can export to *RGB* or to *RGBA*. Use *RGBA* to create screenshots with an alpha layer, enabling transparent backgrounds.

Recompute Alpha Layer. Force alpha layer to be recomputed. This is a costly process. Use only if you have alpha problems in *RGBA* mode.

Full Fame Preview. You can look at the full name preview to check if everything is correct. Note that the resolution information used for the full name preview is the one of the first resolution in the list.

3.4 Resolutions

Managing resolutions. Use the resolution list to specify which resolutions are going to be captured. Use the list + button to add a new resolution. Select the resolution to be removed and press the list − button, or right click on the resolution and select *remove item element*.

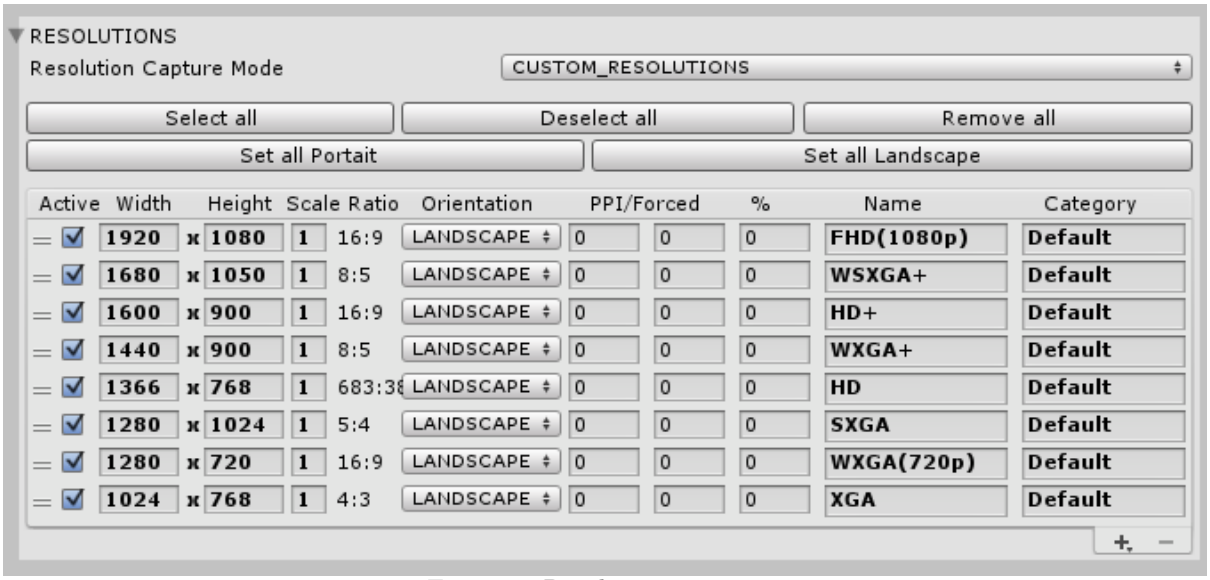

Figure 4: Resolutions settings.

Scale. This setting allows you to easily take ultra HD screenshots. For instance, a resolution of 1600×1200 with a scale of 2 will take a screenshot with a resolution of 3200×2400.

Note that Unity limits the maximum resolution size to $\sim 8000 \times 4000$

Orientation. Switch between *LANDSCAPE* and *PORTRAIT* mode.

Resolution Name and Category. The resolution name can be customized, and used in the file name field with the symbol $\{name\}$. You can also specify a category to be used with the file name symbol $\{category\}$.

Resolution Capture Mode. There are two resolution capture modes :

- *GAME_VIEW* just capture the game view.
- *CUSTOM_RESOLUTIONS* allows you to capture multiple resolutions in one click.

3.5 Cameras

Camera Capture Mode. There are two camera capture modes :

- *GAME_VIEW* just capture the game view.
- *CUSTOM_CAMERAS* allows you the customize the cameras to be used for the capture.

Managing Cameras. Use the camera list to specify which cameras are going to be used for the capture. You can select multiple cameras at a time if your game requires a layered camera rendering (for instance for UI elements or overlays, or if you have a camera for the scene rendering and an other camera for rendering the player gun).

Note that the other cameras of the scene will be disabled during the capture.

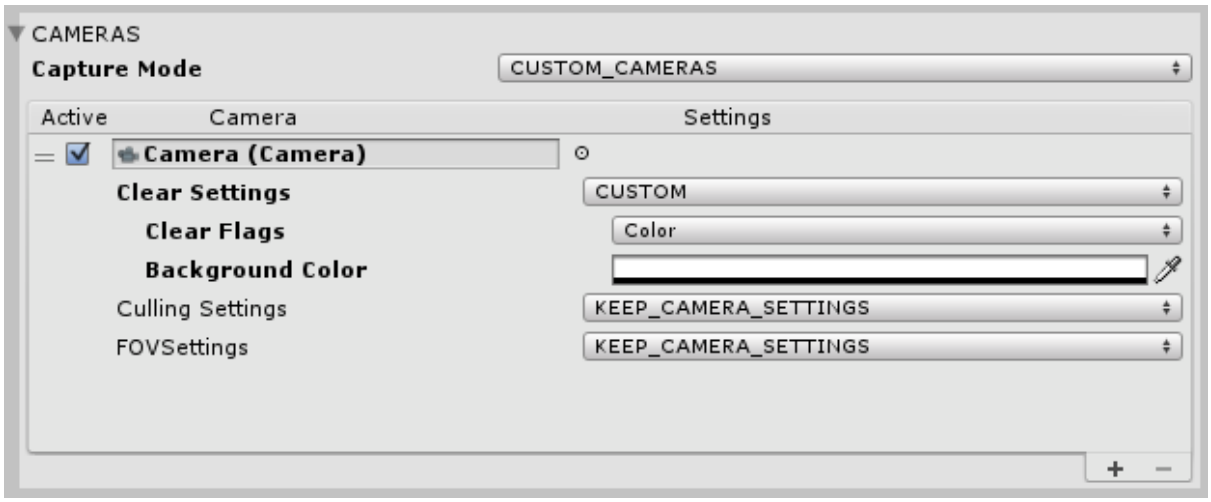

Figure 5: Camera settings.

Custom Camera Settings. Camera settings can be overwritten during the capture:

- *KEEP_CAMERA_SETTINGS* keeps the camera settings unchanged,
- *CUSTOM* allows you to change the camera clear mode and background color, the camera culling mask, and the camera Field Of View.

Note that the custom settings override the current camera settings during the capture process, and should correctly restore them at the end of it.

3.6 Overlay Canvas

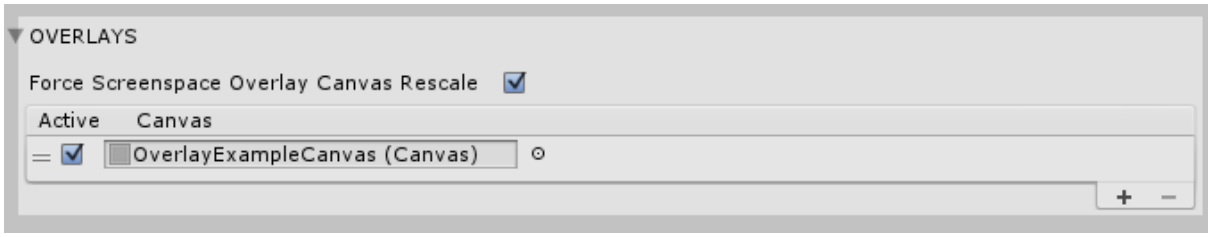

Figure 6: Overlay settings.

The overlay system allows you to easily integrate your game or studio logo into each screenshot, and much more. Use the overlay list to specify which overlays are going to be used for the capture.

Force Screenspace Overlay Canvas Rescale. Force Screenspace Overlay Canvas to be rescaled in *RENDER_TO_TEXTURE* mode.

Render Active UI Canvas. You can choose to hide or not all your active game UI during the capture.

Default Overlay Customization. The *OverlayExampleCanvas* is a *Canvas* that can be found in the children list of the *ScreenshotManager* prefab. By default the game object is inactive, and will be activated only during the capture. You can customize it or create your own overlay canvas.

Creating your own overlays. The overlay system is based on the Unity 4.6 Canvas system, and allows unlimited customization of the screenshot overlays. To create an overlay:

1. Create a new Unity *Canvas* as a child of the *ScreenshotManager* prefab.
2. Edit your new *Canvas* as you want.
3. Create a new overlay item in the screenshot manager list, and select your new *Canvas*.
4. Disable the *Canvas* game object. This keeps the *Canvas* hidden during the game and only displays it during the capture.
5. Save the *ScreenshotManager* prefab or create a new one to save your modifications.

Figure 7: Example of screenshot captured using the default overlay (Viking scene, ©Unity).

3.7 Preview

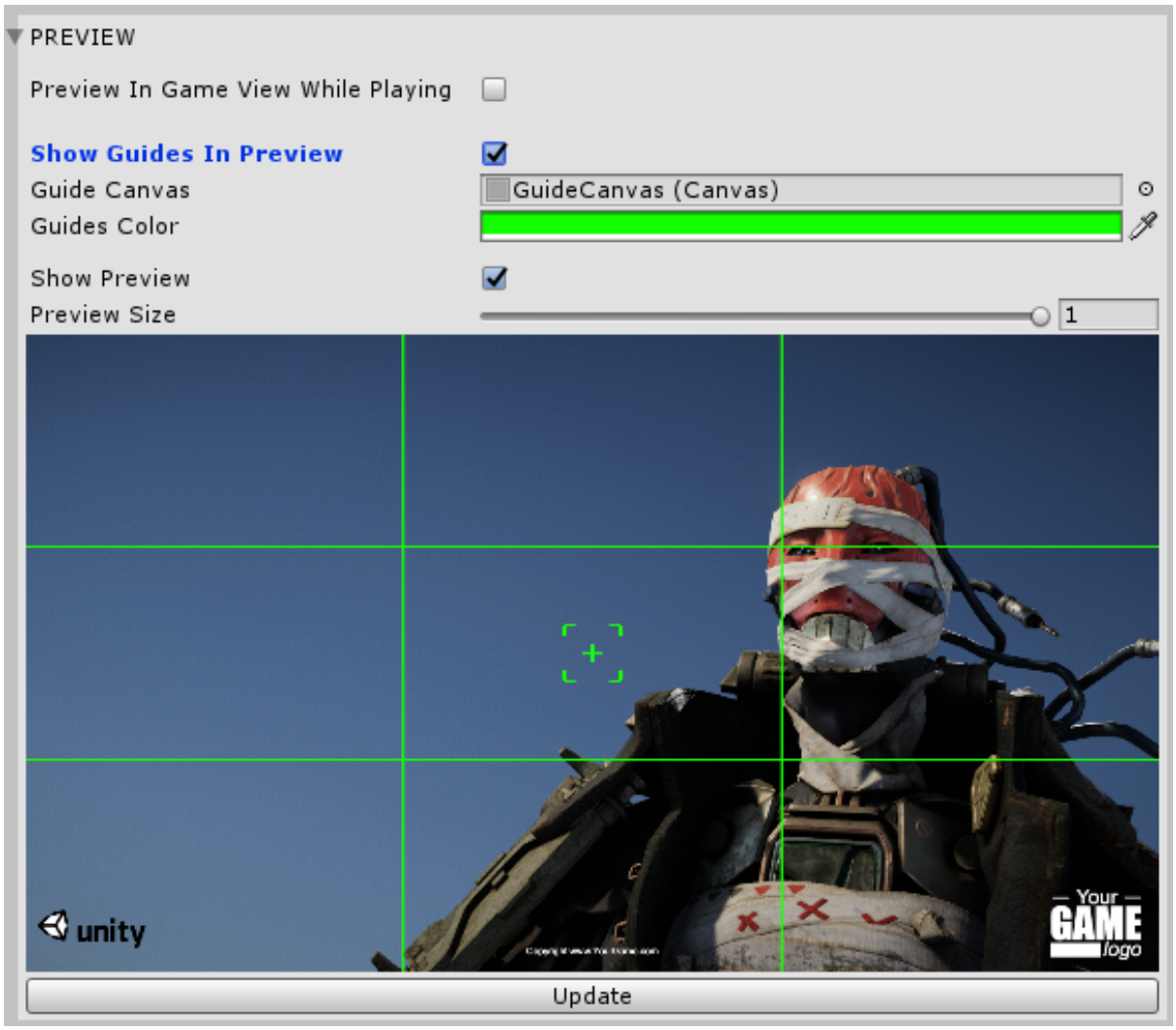

Figure 8: Preview settings. (Character from Adam, ©Unity)

Guides. You can display a photography guide to better frame your screenshots. The default guide is a *Canvas* than can be found in the *ScreenshotManager* children. This guide is visible in the preview picture only, and will not be rendered in the final image.

Preview in GameView while Playing. If you want to visualize how the screenshot will look while playing, you can set this option to true. When the game will start playing, the preview settings will be applied to the GameView, and restored when the game is stopped.

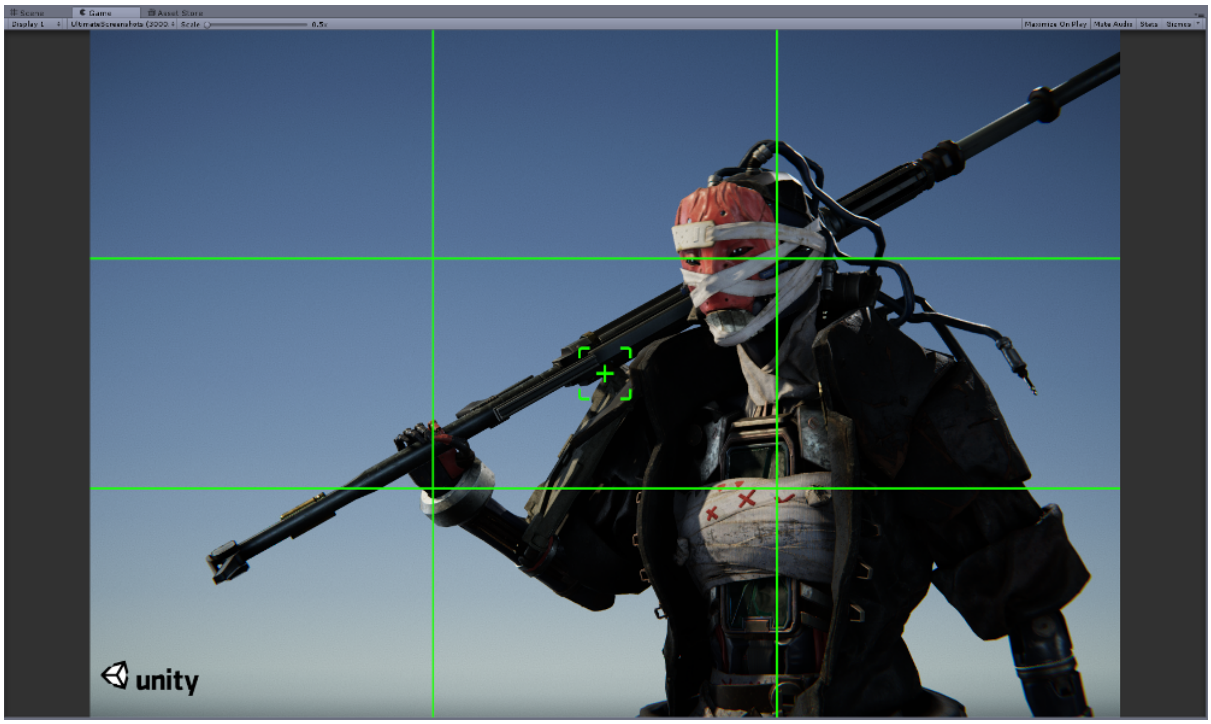

Figure 9: Use the preview in GameView while playing to better frame your screenshot. (Character from Adam, ©Unity)

3.8 Capture

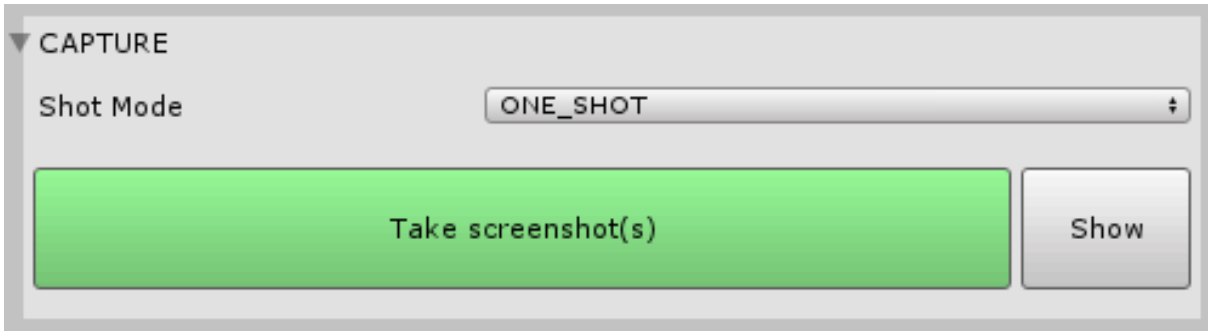

Figure 10: Capture settings.

Capture Mode. There are two capture modes:

- *ONE_SHOT* captures one frame,
- *BURST* allows to take a series of screenshots with a fixed and customizable timestep. Note than you can use the screenshots as inputs of a GIF creator software.

3.9 Utils

Align to View. You can align all the cameras in the camera list with the current scene view. Note that this is a reversible operation using Undo/Redo.

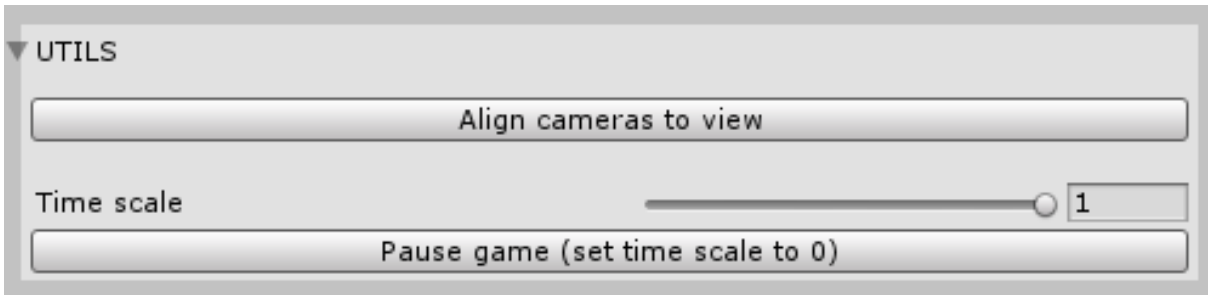

Figure 11: Utils.

Time Scale. You can use these tools to pause or slow down the time while playing to edit the scene and/or better frame your screenshot.

3.10 Misc.

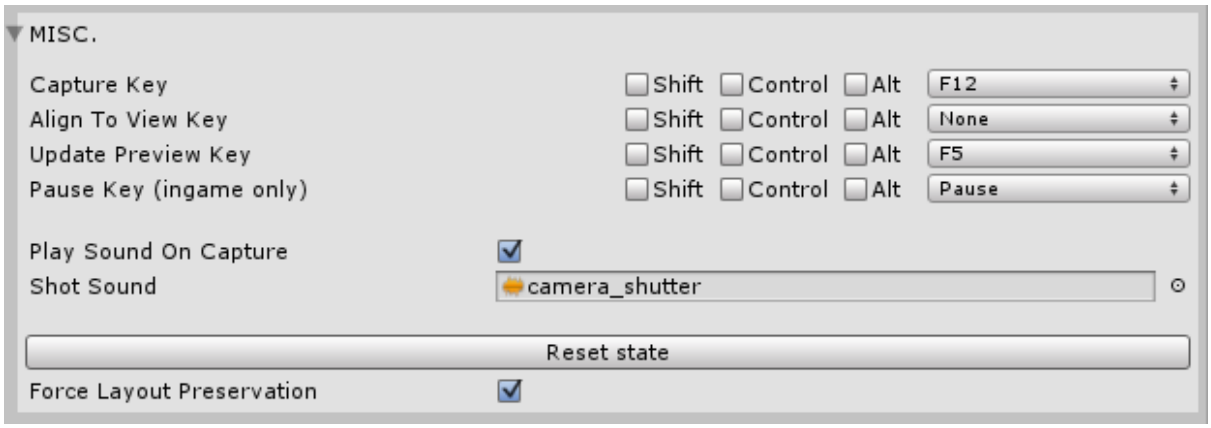

Figure 12: Misc.

In this panel you can set the various hotkey values, and specify the sound to be used for the capture.

Note that these hotkeys only work in play mode.

Reset State. See FAQ 5.8.

Force Layout Preservation. See Faq 5.7.

4 PROGRAMMING

4.1 API

4.1.1 Screenshot Taker

You can capture a screenshot using only the Screenshot Taker component.

Captures the current screen at its current resolution, including UI.

```
public void CaptureScreen(string fileName,
                          TextureExporter.ImageFormat imageFormat =
                              TextureExporter.ImageFormat.PNG,
                          int JPGQuality = 100,
                          bool renderUI = true,
                          bool playSound = false,
                          ColorFormat colorFormat = ColorFormat.RGB,
                          bool recomputeAlphaMask = false)
```

Capture the scene with the specified width, height, resolution upscale, and export it to the file fileName, using the mode *RENDER_TO_TEXTURE*. No UI is captured with this mode.

```
public void CaptureScene(int width, int height, int upscale, string fileName,
                         List<Camera> cameras,
                         int antiAliasing = 8,
                         TextureExporter.ImageFormat imageFormat =
                             TextureExporter.ImageFormat.PNG,
                         int JPGQuality = 100,
                         bool playSound = false,
                         ColorFormat colorFormat = ColorFormat.RGB,
                         bool recomputeAlphaMask = false)
```

Captures the specified resolutions with custom parameters.

```
public void Capture(List<ScreenshotResolution> resolutions,
                    List<ScreenshotCamera> cameras,
                    List<ScreenshotOverlay> overlays,
                    CaptureMode captureMode,
                    int antiAliasing = 8,
                    bool export = true,
                    TextureExporter.ImageFormat imageFormat = TextureExporter
                        .ImageFormat.PNG,
                    int JPGQuality = 100,
                    bool renderUI = true,
                    bool playSound = false,
                    ColorFormat colorFormat = ColorFormat.RGB,
                    bool recomputeAlphaMask = false)
```

4.2 Delegates

If you want to run some custom code before and after the capture processes, you can use the following delegates:

4.2.1 Screenshot Manager

```
public static void onCaptureStartDelegate ()
```

Is called when the capture starts.

```
public static void onCaptureEndDelegate  ()
```

Is called when the capture ends.

4.2.2 Screenshot Taker

```
public static void onResolutionUpdateStartDelegate (ScreenshotResolution res)
```

Is called just before capturing the given resolution.

```
public static void onResolutionScreenResizedDelegate (ScreenshotResolution res)
```

Is called just after resizing the gameview in *GAMEVIEW_RESIZING* mode.

```
public static void onResolutionUpdateEndDelegate (ScreenshotResolution res)
```

Is called just after capturing the given resolution.

```
public static void onResolutionExportSucessDelegate (ScreenshotResolution res)
```

Is called just after the screenshot is exported.

```
public static void onResolutionExportFailedDelegate (ScreenshotResolution res)
```

Is called just after the screenshot export fail.

5 FAQ

5.1 How to capture only a sub-part of the screen

If you want to capture only a sub-part of the screen, add the *ScreenshotCutter* component to one of your scene object. Set the SelectionArea property to the part of the screen you want to capture. It can be any RectTransform, such as an image or a UI panel. The *CutterExample* scene and *ScreenshotCutterExample* prefab illustrate a possible use of the screenshot cutter.

Note that the screenshot cutter only works on *GAMEVIEW_RESIZING* or *FIXED_GAMEVIEW* modes.

5.2 How to Hide Some Scene Objects

Using a component You can hide a specific objet by adding the component *HideOnCapture* to the game object.

Using a culling layer You can hide all objects of a specific layer in *CUSTOM_CAMERAS* mode:

1. For each camera, select the camera and *CUSTOM_SETTINGS* culling mode.
2. For each camera, deselect the layers to be hidden in the culling list.

5.3 Temporal anti-aliasing, special effects artefacts or UI scale issues

In editor, using the *GAMEVIEW_RESIZING* mode, if you have any artefact when taking a screenshot at a custom resolution, or a UI element at the wrong size, you can increase the *ScreenshotTaker m_GameViewResizingWaitingFrames* value. The *ScreenshotTaker* component is located on the same game object that the *ScreenshotManager*.

The *m_GameViewResizingWaitingFrames* value specifies how many frame the screenshot taker waits before to take the screenshot after the gameview has been resized. The default value of 2 should be enough for most settings. Increase this number when some elements are not well updated, like GUI, or when you see some post effects artefacts. Post effects like temporal anti aliasing requier a value of at least 10.

5.4 How to get the screenshot textures

The textures can be accessed using the *ScreenshotManager GetActiveResolutions()* method, and then access their *m_Texture* field.

You can also access to the last taken texture using the *ScreenshotTaker m_Texture* field.

5.5 How to display a preview ui before saving the screenshots

The *ValidationCanvas* script shows how to call a capture event, and then display a validation ui to save or discard the images. An example is available in the example scene *SceneExample*, with the game object called *ValidationGUIExample*.

The idea is as follow:

1. Update the texture without exporting them using *UpdateAll()*.
2. Wait for the capture end event.
3. Display a UI and update its texture by accessing the screenshot texture.
4. Call *ExportAllToFiles()* if the user validates the screenshot.

5.6 How to Create a Screenshot with a Transparent Background

In $CUSTOM\_CAMERAS$ mode:

1. Select the first camera, and set $CUSTOM\_SETTINGS$ clear mode.
2. Select the *color* clear mode.
3. Set the custom color and set its alpha to zero.
4. Set the export format to PNG and choose the $RGBA$ color format.
5. If you have an issue with the alpha layer, try the *RecomputeAlphaLayer* settings.

5.7 My GameView and layout are doing strange things when I capture a screenshot

This can occur when you use the $GAMEVIEW\_RESIZING$ capture mode.

For Unity 4.6 to 5.3, the algorithm rescales the GameView window to match the screenshot dimensions, and this undocks the GameView window. To prevent it, the editor layout is saved before the capture and restored after it, creating a sort of "flashing" effect. If this annoys you, you can set *Force Layout Preservation* to false.

With Unity 5.4 and later, the GameView has an inner "scale" which allows the modification of its dimensions without modifying the editor layout. During the capture, its resolution is changed several times before being restored.

5.8 Nothing Happens when I try to Update the Preview(s)

Sometimes, the renderer may be locked and refuse to update the preview. This happens rarely, but if you do not have any response from the *ScreenshotManager*, click on *Reset State* button.

Do not hesitate to contact me if you can reproduce this bug, so I can correct it.

6 VERSIONS

1.4.3 - 09/01/2018

- Support of Unity 2017.3.
- New feature: validation canvas to preview the screenshot before saving.
- You can now request the iOS gallery authorization when you want.
- Added the possibility to increase the number or waiting frames in $GAMEVIEW\_RESIZING$, to prevent post effect artefact like temporal anti aliasing.
- Updated documentation and FAQ.
- (Fix) WebGL plugin error.

1.4.2 - 01/11/2017

- (Fix) GameViewResizing in NET 4.6.

1.4.1 - 26/10/2017

- Support of Unity 2017.2.
- Cosmetic update: logo updated.
- (Fix) File name update in burst mode.

1.4.0 - 17/10/2017

- New feature: screenshot cutter, capture only a sub-part of the screen.
- Added support to export to secondary storages on Android.

1.3.0 - 27/09/2017

- Better API for developers.
- (Fix) iOS picture export folder.

1.2.2 - 12/09/2017

- Added support for WebGL.

1.2.1 - 29/08/2017

- (Fix) Android picture export folder.

1.2.0 - 16/08/2017

- Support of Unity 2017.1.
- New capture mode: $FIXED\_GAMEVIEW$ to capture the game screen on all platforms, including UI.
- New export mode: $PICTURE\_FOLDER$, to export in the plaform specific picture folder.
- Automatically export the screenshots to the Android gallery.
- Automatically export the screenshots to the iOS gallery.
- Multi-display support.
- Added a message canvas do display a success or failure text in-game.

1.1.2 - 26/04/2017

- Fixed inspector background color in Unity Pro.
- Improved example camera controller.

1.1.1 - 03/04/2017

- Support of Unity 5.6.
- MultiSampling AntiAliasing support in $RENDER\_TO\_TEXTURE$ mode.

1.1.0 - 23/03/2017

- New capture mode: $RENDER\_TO\_TEXTURE$.
- Possibility to capture screenshots in builds on all platforms using $RENDER\_TO\_TEXTURE$.
- Orientation is now a ScreenshotResolution property.
- Extracted code from ScreenshotManager.cs to ScreenshotTaker.cs for all capture related code, and to TextureExporter.cs for all export related code.
- Added the possibility to recompute the alpha layer in RGBA if destructed by camera effects.
- Programmer friendly API, can now call a capture from code.
- Tooltips added for the inspector.

1.0.0 - 08/03/2017

- First public release.

7 KNOWN ISSUES

- Possible incompatibility with obfuscators. Whitelist the namespace AlmostEngine.Screeenshot.

8 SUPPORT

Please do not hesitate to contact me if you have a feature request, or any question, issue or suggestion : support@wildmagegames.com.