Post-Processing NPR Effects for Video Games

Milán Magdics
Universitat de Girona
Girona, Spain
magdics@iit.bme.hu

Catherine Sauvaget
Universitat de Girona
Girona, Spain
catherine.sauvaget@gmail.com

Rubén J. García
Universitat de Girona
Girona, Spain
rgarcia@ima.udg.edu

Mateu Sbert
Universitat de Girona
Girona, Spain
mateu@ima.udg.edu

Abstract

This paper describes different interactive, non photorealistic techniques which can be easily applied to video games. Based on a study of art and games, an emphasis is put on considering NPR tools as basic elements to provide different styles and moods, to convey different emotional and experiential representations of a game. We restrict ourselves to screen based effects, which permits any existing game to use our framework with practically no integration cost. This allows us not only to comply with user preferences in rendering style, but also the creation of multiple gaming experiences out of the same game. We show the resulting effects in an in-house videogame and in standard Unity demos, and show how the users can change the style of the videogame by means of a menu.

CR Categories: I.4.3 [Image Processing and Computer Vision]: Enhancement

Keywords: NPR, Video Games, Art, Perception, Art and science.

1 Introduction

Nowadays, advertising, movies, illustrations, video games, images in general are invading our lives. We use images to transmit a message which can be informational or emotional. Realistic images often contain too many details disrupting the information to be delivered. Thus, artists often simplify images highlighting information using an image stylization. The main reason of the growing use of stylization consists in giving sense to these images by highlighting important elements to deliver the desired message. In computer graphics, this can be realized in the Non-Photorealistic Rendering (NPR) field also called expressive rendering. It tries to combine art and science and is based both on technical and result quality. The important point consists in choosing the best effects corresponding to the message we want to deliver [McCloud 1994]. Thus, NPR needs that artists and scientists work together to solve problems related to graphics. While an artist uses tools or create a painting, we have to generate automatically these tool effects or the style of the painting. However, defining a style is not easy because there is no precise definition for a given style which could be transcribed to computer science. Some researchers have tried to define style [Willats and Durand 2005] by describing different systems and finally the high level question which remains is how to choose the line, detail and abstraction level depending on the message we want to deliver.

The answer, however, may vary in interactive applications from time to time and scene to scene with the dynamic change of the virtual environment. As a consequence, interactive applications such as video games may be equipped with an arsenal of styles to support the requirements of changing abstraction levels. Information transfer, on the other hand, might not be the only goal of visual communication, it can also serve as an attractive tool for product selling which is especially true for video games.

An image style, irrespectively of whether it is realistic or stylized, is transformed into the language of computers by defining the building blocks of the programmable rendering pipeline and feeding them with input. The graphics pipeline basically consists of two main steps: changing and projecting the scene geometry into a 2D image while performing lighting calculations, and to modify pixel colors as regular image processing. The first step operates on the so-called *object-space*, the geometric and material description of virtual objects, which gives a high freedom to define the final style of specific objects. On the other hand, this freedom also brings a good deal of tedious work when the style has to change for the entire image, since modifications have to be applied to many different objects. Contrarily, *screen-space methods* have 2D image(s) as their input (might as well including limited geometry information in G-buffers) meaning that the overall looking of the rendered images can be modified with relative ease at the expense of limited freedom for selective stylization. Nowadays, most game engines support post processing effects, turning screen space methods into a cheap and powerful tool for stylization of both under development and existing games, which served as the motivation for our study.

This work addresses the topic of seamless integration of styles into interactive graphical applications by gathering a collection of screen space stylization methods that can be applied on any existing video game. We applied these effects on games created with a well-known game engine, Unity [Unity Technologies 2013], in a non-invasive manner. Our work is also motivated by the ability to replay a game and the pleasure to rediscover it by changing the stylization and/or the mood. This principle is in the same way as the interesting and well know possibility to change the difficulty level. Our paper is organized as follows. In Section 2, we present different artistic principles from contours to colors, followed by the discussion of the related work in computer graphics and a review of video games and films using NPR methods in Section 3. Next, section 4 summarizes the main characteristics of our target platform: the Unity game engine. Section 5 offers a collection of effects suitable for real-time applications. We present our results in Section 6, including tests in different games, performance analysis and user study. Finally, we conclude our paper in Section 7.

2 Art and Perception

We are surrounded by tones and contrast variations, thereby, evolution has made sensitivity to contours (i.e. abrupt changes in the level of light intensity) the cornerstone of our visual perception [Marr and Hildreth 1980; Koenderink 1984; Palmer 1999]. Contours are also present at the higher levels of perception: when they are not drawn, the human visual system tends to imagine these contours to dissociate the different objects observed. To do that, our brain reduces the number of information keeping only the basic ones. Artists have understood that drawing contours is useful to express ideas and emotions and that shape and inclination of a line can provide a large range of emotions [McCloud 1994]. Contours may serve two purposes: they are, by themselves capable of depicting abstracted objects that our brain is able to recognize, and to aid our perception process by enhancing visual details (e.g. silhouettes or texture lines) on an image with great diversity in color and texture (Figure 2).

The principle is similar for colors. The more numerous the colors are, the more different our respective perception can be. However, even if our perception is different, we usually agree on some common symbolism that permits to create atmospheres which can cause specific psychological state. For example, dramatic atmosphere can be produced with brown and dark blue; green and dark blue denote mystery; violence is usually represented by orange, red and fire colors when threat is depicted by ochre and brown. Sweet green or blue and pink harmonies can convey a sentimental mood [Duc 1983]. Figure 1 shows two images in which grey tones are used to highlight the red parts referring to the blood. In the second image, the brown colour adds to the creepy atmosphere.

Figure 1: *Left: drawing made by Pierre Le Pivain aka Le Pixx. Right: drawing made by Alain Brion. Images used with permission.*

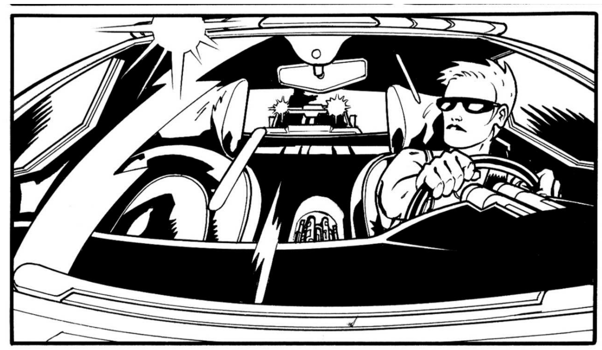


Figure 2: *Comic panel made by Pierre Le Pivain aka Le Pixx. Image used with permission*

Figure 3: *Top: Illustrations by Alain Brion: details disappear with depth, and desaturation is used at the background. Bottom: The Meadow of San Isidro on his Feast Day by Francisco de Goya: Desaturation and lack of details in the foreground; desaturation in the background, and darkening in the middle ground. Images used with permission.*

Ambience is also created by the light and shadows. A very common stylization of comics can be seen in Figure 2, shadows are represented using hatching or black flat areas. Lights and shadows may also be a great help in understanding scene geometry and produce different sensations of depth. For example, an oblique light (45 degrees) produces an ideal depth perception while a front light removes it completely [Roig 2010].

Colors also permit the creation of the illusion of depth. Illustrators and designers often use the detail level and the desaturation of colors for this purpose (see Figure 3). They also emphasize depth with the decreasing thickness of contours at the background (see Figure 2).

Another aspect of visualization is the viewpoint and perspective. Rob Pepperell discusses in [Pepperell 2012] the problem of how to depict the relative indistinctness of peripheral vision as compared with central vision, and the appearance of our bodies in our field of view. He comments that when artists like Paul Cézanne and Vincent Van Gogh attempted to depict the visual periphery they did so not with blurriness but by passages that became increasingly textured and indistinct moving away from the centre, a kind of painterly scrambling. He adds that in painting, cinema, photography, and computer graphics our images of the world are constructed as if we were looking through an invisible window or frame, which we as observers are "outside" looking in.

The currently applied set of these depiction tools determines the actual style and abstraction level of an image.

3 Related Work in Video Games, Video Processing and Films

Screen space post processing effects, such as bloom, tone mapping or lens flare are part of every decent game engine, with easy application and combination. Surprisingly, despite the great diversity of effects developed by researchers so far, game engines contain almost no screen space NPR effects at all. Video games using non-photorealistic effects had to create their style from the very beginning of the game development process. Good examples of this are the comic-like *XIII* and *MadWorld*, or *Okami*, a video game made in the style of Chinese paintings.

NPRQuake [Bakke et al. 2000] is a version of Quake in which the authors tried to develop a game engine where the user is able to create NPR rendering styles in a non-invasive manner. However, given the framework of NPRQuake, it is only possible to stylize a part of it.

Two video games using NPR to help the user to perceive interesting objects or danger are Tomb Raider 2013 and Assassin's Creed 2. The survival instinct (highlighted parts) of Lara Croft can be used to detect walls she can cling on. In Assassins Creed 2, the eagle vision is used to detect enemies (in red), allies (in blue) and the target (shining).

Also, some online games like Second Life propose to the user the possibility to change the mood of the scene in a specific menu with predefined styles or changing each parameter (water color, sky color, cloudyness...).

In some films, similar NPR techniques have been used in post-processing to change their appearance. Since there are no real-time restrictions, more complex algorithms can be used. Examples include Alois Nebel [Lunák 2011], A Scanner Darkly [Linklater 2006] and Rennaisance [Volckman 2006]. Recently, time-coherent abstraction techniques have been demonstrated on low resolution video [Song-Hai et al. 2011]. They use face detection to detect important areas and guide the edge detection thresholds. Optical flow provides coherency in the segmentation labeling. Example images or videos can be used to set the color palette. 352x288 video (at 10 fps) can be processed in real-time, providing a cartoon-like appearance.

4 Implementation Framework

Since current computer games are becoming quite complex software, we have designed our effects so that they can be used with only small modifications to the games. The effects are created as post-processing effects running mostly on graphics processing units, in order to provide easy portability to different graphic engines and architectures. In particular, we have integrated these effects in the Unity game engine [Unity Technologies 2013]. In Unity, a game is defined as a set of scenes in which different graphic objects are created, evolve and are destroyed according to user input, behaviour scripts and physics. Objects can be linked in a hierarchy tree to create compound objects, or stored as a *prefab* for easy cloning. The game may be run inside the Unity editor, or compiled into a standalone program. Unity supports creating Windows, OSX, Android, IOS or Flash applications targetting PCs, mobile phones or web.

One of the possible types of scripts for cameras is a post-processing effect, which renders a full screen quad on top of the current view, using GPU shaders. The previous contents of the frame buffer and other related information (textures, depth and normal map etc.) are available, and these effects can be chained. Each of our effects consists of either one parametrized script (with the required textures and GPU shaders) or a *prefab* containing the different objects required for rendering more complex effects. The effects have been classified in a coherent library (a.k.a. *unitypackage*), which can be imported into other Unity projects. The procedure to add our effects to a new game basically consists of importing our NPR library of effects, dragging the correct effect script or *prefab* to the active camera in the scene, and setting the tunable parameters in the object inspector. Multiple effects can be added, although in some cases care must be taken to use the correct ordering.

5 NPR Effects for Games: Our Case Study

In [Durand 2002], F. Durand explains that "the relation between the object-space scene and the 2D picture can be quite complex, and that picture generation is not unidirectional, but involves many back-and-forth exchanges, feedback, constraints, and goals linking the scene and the picture. This is related to the complexity of the human visual system, and to the dual nature of pictures, both flat objects and representation of an objective scene." Screen space methods commonly operate on a set of 2D images, including those generated by the rendering engine such as the color, depth or normal buffers and output the final, stylized 2D color image. As previously explained, images have a purpose. Thus, the idea consists in optimizing the image to transmit the desired message. This is the reason why games and softwares should allow users to attempt different stylizations and adjust their preferences to satisfy their requirements. Softwares like Photoshop [1] or Gimp [Gimp 2013] offer many stylization possibilities. However, filters cannot be mixed automatically in the same layer. We think that such filters should not be stand alone application but easy to blend to be convincing.

Thus, we propose in this section, a variety of screen space methods used to modify the style of the rendered image stream as post processing, suitable for video games which implies the requirements of real-time rates and temporal coherence.

5.1 Edge Enhancement

In order to convert an image into line drawing or enhance details, object silhouettes or texture lines have to be identified first. Technically speaking, a so called *edge map* is generated where each pixel denotes whether the corresponding pixel of the color image is edge or not, which can be either a binary value or a floating point value denoting the edge strength, creating smoother lines.

5.1.1 Geometric vs. Texture Edges

There are two main approaches for edge extraction. The first class of methods tries to mimic artists: lines follow object silhouettes and main geometric features, ridges and valleys, resulting in object space methods. However, through the use of depth and normal maps, screen space methods may perform limited object space calculations. A common technique is to apply standard edge detection operators on the depth and normal maps to obtain geometric contours. The second approach follows a perceptual viewpoint: the sensitivity of our eyes to abrupt changes in light intensity, utilizing classic edge detection methods of image processing searching for pixels of high gradient magnitude. While the first class relies on geometry information and thus better depicts shapes, it completely forfeits texture information. In contrast, standard edge detection methods provide a less clean representation of shape contours, but texture details are preserved. The combination of both approaches preserves only their benefits [Redmond and Dingliana 2009a].

[1] Adobe Photoshop: http://www.photoshop.com

5.1.2 Aesthetic Texture Edges

Among the edge detection methods, variations of the *Difference of Gaussian* (*DoG*) filter proved to give the most aesthetic results [Winnemöller 2011]. From a signal processing viewpoint, DoG is a band-pass filter, approximating the Laplacian-of-Gaussians and has a biological relevance by providing a model to the activation mechanism to certain retinal cells [Young 1987]. The popularity of this method comes from several attributes: in most cases, it creates more connected edges than other methods which is more suitable for illustration purposes; edge thickness can be easily controlled by the kernel size and the variance parameters; and finally, with simple extensions, it can produce a variety of convincing effects and styles by itself [Winnemöller 2011] such as pastel, charcoal or wood carving. In our framework, we adopted the separable, flow-based implementation proposed by Kyprianidis et al. [Kyprianidis and Döllner 2008] and the extension proposed by Winnemöller [Winnemöller 2011], simplifying the parameters into one unified thickness parameter.

5.1.3 Edge Colorization

Edges are enhanced by changing the color of the pixels according to the edge map. Edges are most commonly drawn with ink color (mostly black or blue), but arbitrary color may be used. An interesting, brush stroke effect is achieved by setting a constant background for non-edge pixels while using the color map as edge color (see Figure 6).

5.2 Texture Simplification

Starting with the pioneering work of Gooch et al. [Gooch and Gooch 2001] and DeCarlo et al. [DeCarlo and Santella 2002], researchers have been focusing more and more on one important goal of NPR: creating visualizations which emphasize the most important parts of the scene. Since the real world (and thereby, images produced by realistic rendering) is visually very complex, relevant information content is suppressed by unimportant details. Studies show that automatic *image abstraction methods* involving image simplification steps to reduce detailedness of textures and tone complexity can improve user performance in e.g. recognition or search tasks [Winnemöller et al. 2006].

5.2.1 Detail-preserving Smoothing

These methods aim at removing irrelevant texture details while keeping relevant ones. But what detail is considered as relevant? There is no general answer, it depends on the message we want to communicate. However, as our visual perception is sensitive to high contrast details there is a high correlation between information content and pixels of an image with high gradient magnitude. Automatic methods rely on this observation and dampen weak edges while preserving sharper ones. Recent real-time methods use variants of the bilateral filter [Winnemöller et al. 2006; Kyprianidis and Döllner 2008], the Kuwahara filter [Kyprianidis et al. 2009] or they are based on image flow [Kang et al. 2009; Kyprianidis and Kang 2011]. In our framework we implemented the algorithm of Winnemöller et al. [Orzan et al. 2008] iteratively applying the bilateral filter, since we considered it also capable of producing more cartoon-like images like other methods. In its most common use, the weights of the bilateral filter [Tomasi and Manduchi 1998] is a product of two Gaussians: one is defined in the spatial domain, the other is in the intensity domain. The intensity-dependent Gaussian weight ensures that neighboring pixels placed on the same side of a step-like signal as the center pixel get higher weights while neighbors from the other side of the edge give less contribution to the filter output, better preserving edges. The amount of details preserved is controlled by the variance of the Gaussian functions and the number of iterations the filter is applied. The bilateral filter can be well approximated with a fast separable implementation [Kyprianidis and Döllner 2008] at the gradient and tangent directions. It is applied in the CIE Lab color-space to avoid color-bleeding artifacts [Tomasi and Manduchi 1998], typically iterated $1-5$ times.

5.2.2 Color Quantization

Color quantization reduces the number of colors used in an image. Its most important applications are image compression and displaying images on devices that support a limited number of colors. However, since it applies compression, it simplifies the image as well by further smoothing flat regions making it suitable for image abstraction. We used the luminance quantization method of Winnemöller et al. [Orzan et al. 2008]. This work introduced smooth quantization that can control the transition sharpness between neighboring quantization levels. Transition sharpness may be guided by the image gradient, thereby, sharp transitions are kept around edges while for low-contrast regions sudden changes are spread over a large area making them less noticeable between frames.

A side effect of quantization, especially when it is used in conjunction with smoothing and the number of bins is low (typically $8-10$), is the introduction of fake edges producing a toon shading like effect. This is often more beneficial than problematic, especially when a cartoon-like style is preferred. In fact, image abstraction can turn an image into comic-style [Orzan et al. 2008] preserving original color tone of surfaces, but as opposed to toon shading, object materials do not have to be replaced.

5.3 Shadow Recoloring

5.3.1 Increasing Contrast

As presented in section 2, many kinds of lighting effects exist [Parramón 1987]. Based on art history and on a study of Sauvaget et al. [Sauvaget and Boyer 2010], we propose different effects playing with lighting effects. Artists use light to increase contrast between image areas. For example, Chiaroscuro technique consists in catching the viewer eyes in specific parts of the painting by creating violent contrasts between shadowed and lit parts. Impressionists remove very dark colors in their paintings and represent enlighted objects with pastel colors and smooth contrasts variations whereas shadows are represented by bright colors. One can also notice that shadows are often represented by black flat areas in comics. Sometimes, illustrators use the complementary colors to enhance the contrast between light and shadow.

Considering these descriptions, creating a chiaroscuro consists in modifying the value (HSV model) making the parts of the image in shadow darker and the enlighted parts lighter. Impressionism requires decreasing the saturation and increasing the value to create pastel colors while bright colors are created by increasing the saturation and decreasing the value. For comics stylization, shadow pixels can be turned in black or each pixel can get its complementary hue.

5.3.2 Shadow Extraction

In order to perform color modifications we first determine which pixels are shadowed and create a binary *shadow image*. While a minor modification of the rendering code to catch the moment when shadows are applied on a fragment would also work, here we focus on non-invasive approaches. Furthermore, the invasive approach might become time consuming if the rendering engine does not allow access to shadow maps generated by build-in shaders. We found the naïve non-invasive approach the most flexible, which renders the scene twice with identical projection parameters, once with and once without shadows and then compare the two images in a post processing step. When deferred shading is supported by the rendering engine, we can the eliminate the doubled geometric complexity by sharing the same G-buffers for the two distinct rendering passes. We note that this approach is suitable only for dynamic shadows, as standard pre-computed light maps lose information about light source visibility.

An alternative approach is to use purely 2D image processing methods. These in general rely on the fact that shadowed surfaces are darkened and, thereby, apply intensity thresholding. As an example Sauvaget et al. [Sauvaget and Boyer 2010] compute a global threshold based on the mean and the median light intensity of the image, pixels darker than this threshold are classified as shadows. Note that as scene geometry is not considered lit surfaces made of darker materials may also be selected as shadows.

5.4 Depth Sensation with Varying Abstraction Level

Such as atmospheric effects, fog or depth of field are indispensable for realistic depiction of farther objects, artists have their own ways to create sensation of depth. Since the most of their message is placed in the foreground and farther objects serve as the background of the main action with less important details, abstraction level increases with the distance from the viewpoint. The relation between depth and abstraction is guided by two parameters: a linear scaler (to set the focus level) and an exponential coefficient (to set the change rate). Depth quantization is also employed to create a layered-like look. This section shows a few examples for reducing scene complexity by the depth with changing the main components of a style: color, texture and contours.

5.4.1 Desaturation

Artists often use less saturated colors to distract the viewer's attention from less important parts of their images. It was shown by Redmond et al. through both task-based and eye tracking experiments that desaturation can indeed attract the user's gaze [Redmond and Dingliana 2009b]. An important benefit of desaturation comparing to smoothing methods is that only richness in color shades are lost, but most high contrast details including silhouettes are preserved, so information about object shapes still exists in the output.

Saturation can be directly modified by converting the pixel color into HSL or HSV color spaces. In our implementation, we allow the modeler to set the depth which is in "focus" (saturation is unaffected), fragments farther or closer from this point are desaturated, based on the linear distance from the focus point. We also used a gamma-correction with user specified exponent to control how fast saturation changes with depth. Additionally, an optional linear quantization is added to simulate a background built up of several layers, commonly seen in comics or cartoons.

5.4.2 Background Atmosphere

Another, commonly used technique by artists is to give an atmosphere to the image by using the same hue for the background, with varying saturation levels [Sauvaget and Boyer 2009]. On the implementation level this can be simply done by specifying the hue for backround pixels in the HSL or HSV color space.

5.4.3 Blur

When also the contrast details of background objects are of less importance, background is often smoothed in artistic depictions. Similarly to detailedness in color, objects richer in contrast details can attract user gaze [Redmond and Dingliana 2009b]. As the amount of details preserved is controlled by the parameters of edge preserving filters (see Section 5.2 and [Redmond and Dingliana 2007]), depth-based parametrization of these filters can produce higher abstraction level for the background objects. More specifically, in the case of the method of [Winnemöller et al. 2006] the variance parameters of the spatial domain and intensity domain Gaussian filters are responsible for the amount of blur and the strength of details preserved, respectively [Tomasi and Manduchi 1998], thus, increasing both by the pixel depth results in an increased abstraction.

5.4.4 Contour Thickness

Contour style, including variations in thickness varies from artist to artist. However, as the study of Goodwin et al. [Goodwin et al. 2007; Hertzmann 2010] showed, the contour thickness drawn by a great fraction of artists shows similarities with the so called isophote distance. Although the formula is computationally too complex to handle it in real-time rates, it also shows that contour thickness for the same objects is inversely proportional to depth, similarly to the method proposed by Gooch et al. [Gooch et al. 1999]. This latter depth dependency is easy to implement, as we used an edge detection method with controllable thickness (Section 5.1).

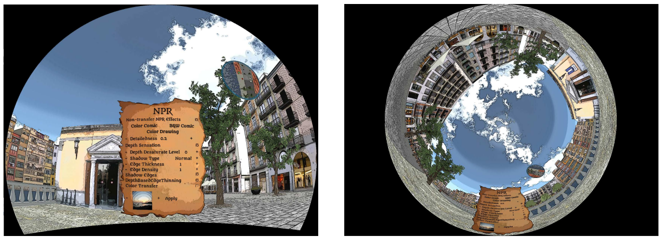
5.5 Color Palette Modification by Example

The color histogram of an image determines its overall mood. Most current game engines are equipped with basic histogram modification methods (e.g. tone-mapping, grey-scale conversion, gamma correction, bloom etc.). However, the possibility to change an image's mood with these tools is very limited. In addition, the naïve definition of a flexible color mapping would require tedious work from the game designers and artists. In terms of human work, one of the most efficient ways to specify a very high dimensional model is *modeling by example*. Automatic *color style transfer* methods follow this idea by letting the user to specify example image(s) and the color palette of the input image is adjusted to have a similar color histogram as the exemplar. Although color transfer has a wide range of literature, due to their high computational complexity only a very few methods are applicable in real-time. We note that only application of the color style has to be real-time, extraction of the style information (e.g. average color) may be performed off-line.

A simple and fast approach was introduced by Reinhard et al. [Reinhard et al. 2001]. This method adapts the average color and variance of the input to the exemplar. Different color spaces such as $L\alpha\beta$ [Reinhard et al. 2001] or oRGB [Zhao et al. 2009] may be used to compute the average and variance to get slightly different results. Pyramid reduction techniques are commonly used as an efficient parallel implementation for computing the average and variance [Szirmay-Kalos et al. 2008].

5.6 Discussion: Rendering Quality and Abstracion

As the reader might have observed so far, none of the effects described in this section are able to enrich the image with new details not contained inherently in the input of the post processing pipeline (except the fake edges of quantization). Edge enhancement, for example, emphasizes changes in local contrast, but does not produce new information. Many effects on the other hand, especially those



(a) (b)

Figure 4: *NPR effects together with post-processing filters for immersive displays, showing the menu to change the effects in-game. The menu will appear in front of the user in the corresponding device. (a) Immersapod (b) Spherical Dome.*

of Section 5.2, may eliminate fine details produced by the rendering pipeline with great effort, which may seem contradictory. Indeed, a similar effect could be achieved by using a simpler description and illumination model of the scene, however, fine tuning the abstraction level in object space would conflict with our goal of producing a set of styles with as few work as possible. Nevertheless, a very beneficial consequence is that a lower quality of rendering settings such as lower geometric level of detail, texture resolution or less accurate illumination model may be used to produce the input of stylization methods saving computation time, since artifacts of an inaccurate model will be blurred anyway. In general, the higher the abstraction level, the lower the rendering quality we may choose. Additionally, it is worth noting that our routines work best for games which provide a realistic rendering of the scenes. If a game has been been carefully designed to elicit a particular mood, this conditions the final result of our routines. For example, a game with a very dark appearance has a reduced palette, so the results of reconstructing a bright, happy scene will not look believable.

6 Results

In order to prove the generality of our routines we have integrated the effects into the video game called "Legends of Girona" we have developed [Rodríguez et al. 2013], as well as into some standard Unity demo projects [Unity Demo Projects 2013]: Car tutorial, Angry bots and 3rd Person Shooter. In each case, integration including the application of the scripts and parametrization to create the desired styles took no longer than a couple of minutes. Figure 5 shows screenshots from the tested games without stylization. Examples for edge detection, texture simplification, shadow colorization, desaturation and color transfer are shown by Figures 6, 7, 8, 9 and 10, respectively and explained in their caption.

We also tested the performance of our effects in the Car Tutorial project on an Intel Core I7-2600 @ 3.4 GHz with an AMD Radeon HD 6900 graphics card, frames per second values of the different effects are shown in Table 1. We can see that most of the post-processing has little effect on performance; however the calculations of shadows and edges can be expensive at high resolutions. A minimum of 30 fps was obtained with all effects active at maximum resolution, which we consider acceptable for the results obtained.

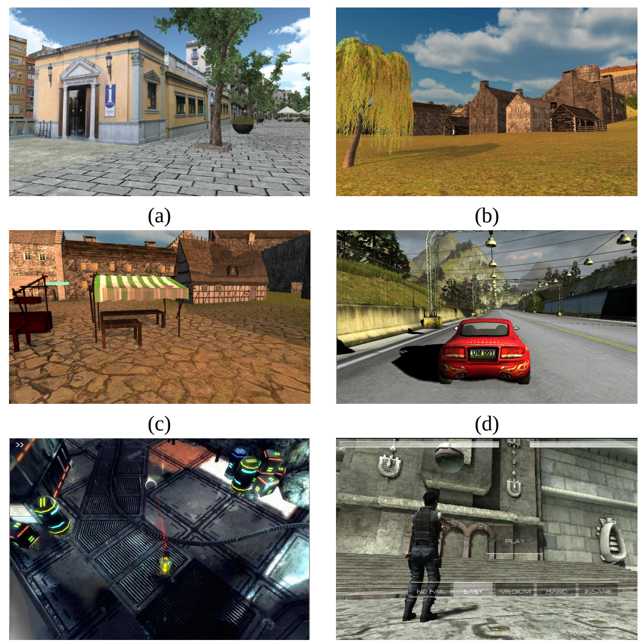
We also tested our effects to ensure that they did not interfere with other screen space effects. Figure 4 shows the effects working in combination with a post-processing filter to visualize the game in Dome and Immersapod immersive devices (details for the Dome filter can be seen in [García et al. 2013]).

Resolution	No NPR	Edges	Shadows
1024x768	60	60	60
1280x960	60	60	60
1600x1200	60	58	41
1920x1080	60	50	39

Resolution	Simplification	Desaturation
1024x768	60	60
1280x960	60	60
1600x1200	58	60
1920x1080	47	60

Resolution	Color transfer	All effects
1024x768	60	60
1280x960	60	59
1600x1200	60	39
1920x1080	60	30

Table 1: *Performance of the different effects. Note that the FPS counter of Unity Car Tutorial used in our tests is limited to 60.*



(a) (b) (c) (d) (e) (f)

Figure 5: *Images of the tested games* without stylization*: (a) Legends of Girona urban scene, (b) Legends of Girona historical scene no. 1, (c) Legends of Girona historical scene no. 2, (d) Car tutorial, (e) Angry bots and (f) 3rd Person Shooter.*

6.1 In-game selection of effects

Most games have a menu to change the rendering parameters to increase or decrease the realism of the visualization, to ensure smooth playing experiences across different CPUs and GPUs. This menu includes the activation of antialiasing, global illumination and shadows, among other parameters, and allows the user to choose his prefered screen resolution.

Following this standard industry practice, we created a menu to control the different aspects of the NPR rendering. With this setting, users can choose their preferred stilization in the same manner they change the difficulty of the game. Since all the parameters of

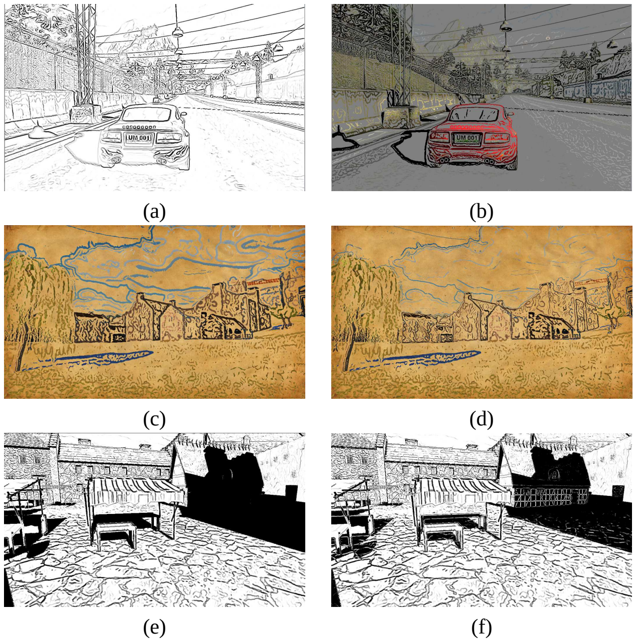


Figure 6: *Various edge enhancement effects, including black and white images and edge drawings guided by the color map. Edge map was produced by the method of [Winnemöller 2011]. The top row shows (a) black and (b) colored edges, with constant edge thickness. The middle row compares (c) global and (d) depth-guided edge thickness, shadows are drawn with complementary color. The bottom row combines black and white edge drawings using depth-guided thickness with (e) black shadows and (f) black shadows with negative color.*

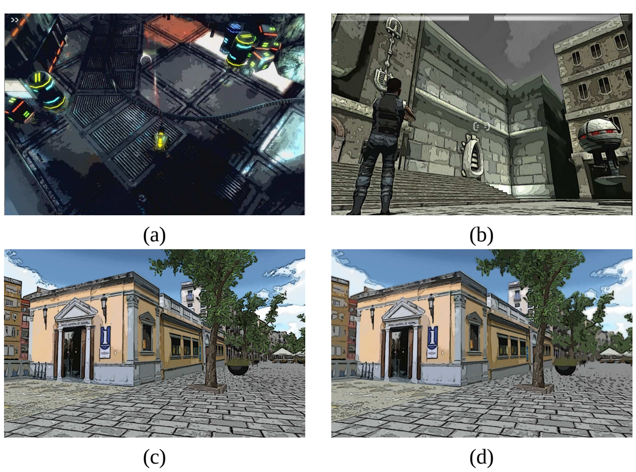


Figure 7: *Texture simplification examples with luminance quantization based on the methods of [Winnemöller et al. 2006] and [Kyprianidis and Döllner 2008]. The bottom row shows the differences between (c) fixed blur and (d) depth-guided-blur, note that in the latter case less and less details are kept farther from the camera.*

the effects can be changed at run-time, users are not constrained

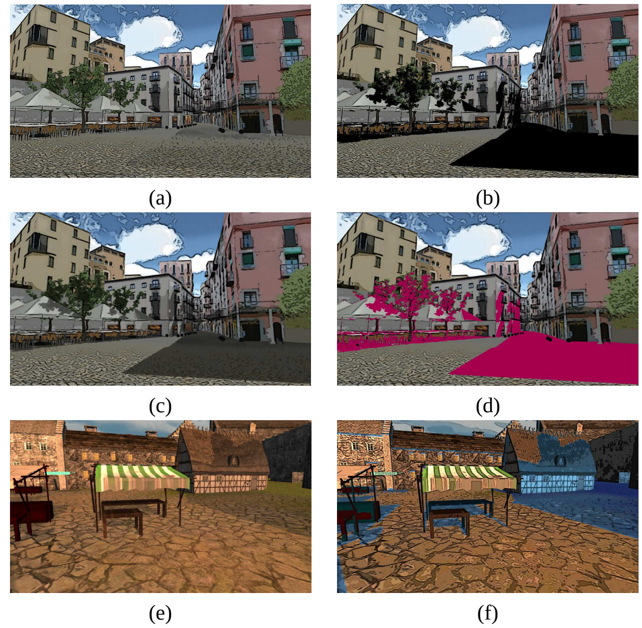


Figure 8: *Shadow effects: (a) built-in shadows of Unity, (b) black shadows, (c) darkened shadows, (d) shadows with constant color. The bottom row compares (e) built-in shadows and (f) complementary color, note that the original colors are mostly yellowish implying a blueish complementary color.*

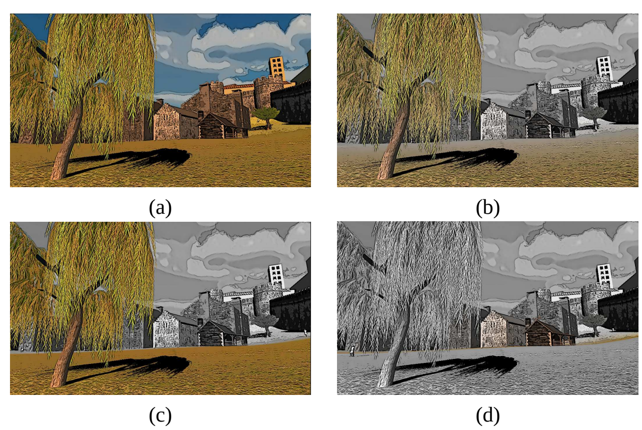
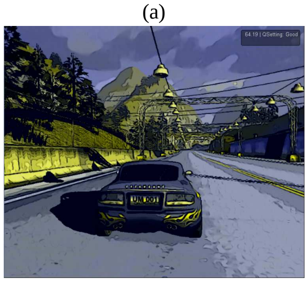


Figure 9: *Desaturation examples: (a) no desaturation, (b) gradual desaturation, (c) desaturation with sharp transition using an exponential function of the depth and (d) focusing on a specific depth level.*

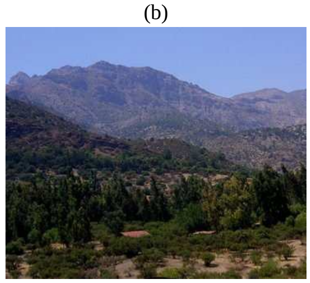
in any way when designing their preferred style. However, we do provide three predefined styles to help in the initial setting of the parameters, and we provide a high-level view of the parameter space to avoid overwhelming the final user. Figure 11 shows a screenshot of the NPR menu, in which changes are immediately propagated to the scene which can be seen in the background to provide visual feedback to the user.

(a)

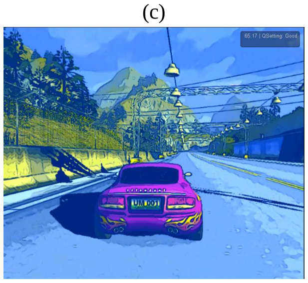
(b)



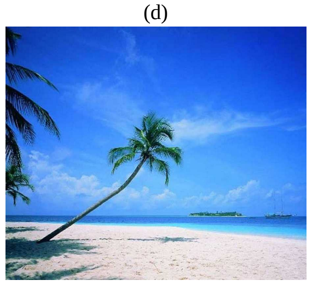
(c)



(d)



(e)



(f)

Figure 10: *Results of color transfer (left column — (a), (c) and (e)) for different example images (right column — (b), (d) and (f)), based on [Reinhard et al. 2001]. As one can see, specific moods are created using this method.*

7 Conclusions

In this paper we investigated some possibilities for changing the rendering style of existing video games by applying post processing filters. In particular, edge enhancement and colorization, texture simplication, shadow recoloring, abstraction-based depth sensation and color palette modification by example have been implemented as GPU shaders to provide real-time performance.

We showed that the discussed methods can be integrated seamlessly into any games with negligible efforts (we used four Unity demos and the Legends of Girona game as testbeds), allowing game developers to easily make use of the recent results of the NPR field and better optimize the appearance of their virtual world for visual communication and entertainment. We also showed how the final users may create their own style using an easy to use menu. We performed an informal evaluation of our system by casual gamers, modellers and computer scientists with background in videogame development, with positive feedback: the effects were found easy to use by modellers and game developers, and aesthetically pleasant by gamers.

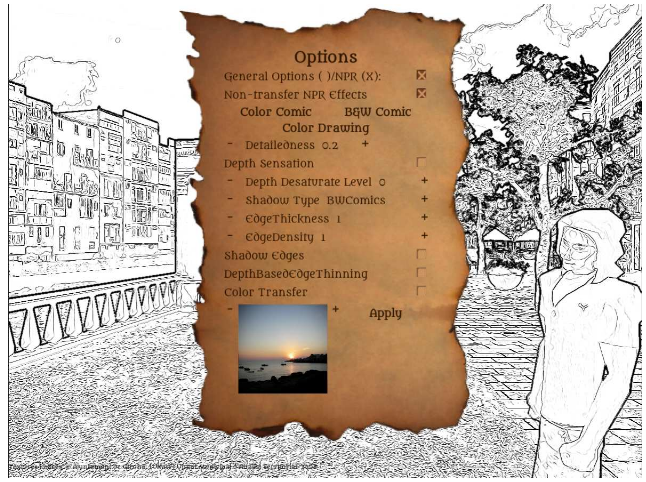


Figure 11: *NPR menu in Legends of Girona.*

8 Future Work

For future work, we plan to perform a user study of the routines, both for game developers and artists, on one hand, and on final users and players, on the other hand, to validate the usefulness of the routines, and to study how each of them affects the player feelings. We would also like to investigate further possibilities of non-invasive post processing effects, including stylizations of shadow shapes, stroke based rendering such as hatching, or the depiction of peripheral vision discussed in Section 2. We will add the possibility of selective stylization to highlight desired parts of an image corresponding to specific meaning and semantics. In a longer term, we plan to create a recommendation system that will help the user to create a style which may depend on the kind of the scene to adapt the style and the atmosphere.

Supporting additional engines, such as VR-Platform [VRPlatform 2013] would also be advantageous. Our routines could also be used with almost any existing videogame without modifying the source code at all, by intercepting the drawing calls and adding the necessary processing. The OpenGL or DirectX library calls can be captured by creating a new library which defines the same functions, and which performs a passthrough by default to the real library. Example libraries using this technique for OpenGL are Bugle [Bugle 2013] and glintercept [glintercept 2013]. Direct3D9Interceptor [Matt 2013] provides a similar functionality for DirectX.

Acknowledgement

This work has been supported by the research projects coded TIN2010-21089-C03-01, IPT-2011-1516-370000 and IPT-2011-0885-430000 (Spanish Commission for Science and Technology), and by grant 2009SGR643 (Catalan Government). We would also like to thank all anonymous reviewers for their comments.

References

BAKKE, E., DUTCHER, S., HERMANN, C., GARDNER, A., AND MOHR, A., 2000. NPRQuake. Computer Science Course Project.

BUGLE, 2013. BuGLe: an OpenGL debugging tool. http://www.opengl.org/sdk/tools/BuGLe/.

DECARLO, D., AND SANTELLA, A. 2002. Stylization and abstraction of photographs. *ACM Trans. Graph. 21*, 3 (July), 769–776.

DUC, B. 1983. *L'art de la BD*, glénat et b. duc ed. Glénat et B. Duc.

DURAND, F. 2002. An invitation to discuss computer depiction. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, ACM, New York, NY, USA, NPAR '02, 111–124.

GARCÍA, R. J., MAGDICS, M., RODRÍGUEZ, A., AND SBERT, M. 2013. Modifying a game interface to take advantage of advanced I/O devices: a case study. In *Proceedings of the 2013 International Conference on Information Science and Technology Application. Advances in Intelligent Systems Reseearch*, vol. 58, 128–132.

GIMP, 2013. GIMP Plug-In Registry. A repository of optional extensions for the Gimp.
http://registry.gimp.org/index.jsp.

GLINTERCEPT, 2013. glintercept - OpenGL call interceptor/logger - Google Project Hosting.
https://code.google.com/p/glintercept/.

GOOCH, B., AND GOOCH, A. 2001. *Non-photorealistic rendering*. A K Peters.

GOOCH, B., SLOAN, P.-P. J., GOOCH, A., SHIRLEY, P., AND RIESENFELD, R. 1999. Interactive technical illustration. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, ACM, New York, NY, USA, I3D '99, 31–38.

GOODWIN, T., VOLLICK, I., AND HERTZMANN, A. 2007. Isophote distance: a shading approach to artistic stroke thickness. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, ACM, New York, NY, USA, NPAR '07, 53–62.

HERTZMANN, A. 2010. Non-photorealistic rendering and the science of art. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, ACM, New York, NY, USA, NPAR '10, 147–157.

KANG, H., LEE, S., AND CHUI, C. K. 2009. Flow-based image abstraction. *IEEE Transactions on Visualization and Computer Graphics*, 62–76.

KOENDERINK, J. J. 1984. What does the occluding contour tell us about solid shape? *Perception 13*, 3, 321–330.

KYPRIANIDIS, J. E., AND DÖLLNER, J. 2008. Image abstraction by structure adaptive filtering. In *Proc. EG UK Theory and Practice of Computer Graphics*, 51–58.

KYPRIANIDIS, J. E., AND KANG, H. 2011. Image and video abstraction by coherence-enhancing filtering. *Computer Graphics Forum 30*, 2, 593–602. Proceedings Eurographics 2011.

KYPRIANIDIS, J. E., KANG, H., AND DÖLLNER, J. 2009. Image and video abstraction by anisotropic kuwahara filtering. *Computer Graphics Forum 28*, 7, 1955–1963. Special issue on Pacific Graphics 2009.

LINKLATER, R., 2006. A scanner darkly. Producer: Warner Independent Pictures.

LUNÁK, T., 2011. Alois Nebel. Producer: Negativ, Česká Televize, Pallas Film.

MARR, D., AND HILDRETH, E. 1980. Theory of edge detection. *Proceedings of the Royal Society of London Series B 207*, 187–217.

MATT, 2013. Matt's Webcorner - Direct3D 9.0 API Interceptor.
http://www.graphics.stanford.edu/˜mdfisher/D3D9Interceptor.html.

MCCLOUD, S. 1994. *Understanding Comics the invisible art*. Harper Paperbacks.

ORZAN, A., BOUSSEAU, A., WINNEMÖLLER, H., BARLA, P., THOLLOT, J., AND SALESIN, D. 2008. Diffusion curves: A vector representation for smooth-shaded images. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008)*, vol. 27.

PALMER, S. E. 1999. *Vision science : photons to phenomenology*. MIT Press, Cambridge, Mass. Stephen E. Palmer.; "A Bradford book."; Bibliography: Includes bibliographical references (p. [737]-769) and indexes.

PARRAMÓN, J. 1987. *Ombres et lumières dans le dessin et la peinture*. Bordas.

PEPPERELL, R. 2012. The perception of art and the science of perception. In *Proc. SPIE 8291, Human Vision and Electronic Imaging XVII, 829113*.

REDMOND, N., AND DINGLIANA, J. 2007. Adaptive Abstraction of 3D Scenes in Real-Time. In *EUROGRAPHICS '07: Proceedings of the Short Paper Programme*, 77–80.

REDMOND, N., AND DINGLIANA, J. 2009. A Hybrid Approach to Real-Time Abstraction. In *Proceedings of Eurographics Ireland, 2009*.

REDMOND, N., AND DINGLIANA, J. 2009. Influencing user attention using real-time stylised rendering. In *EG UK Theory and Practice of Computer Graphics, Cardiff University,, United Kingdom, 2009. Proceedings*, Eurographics Association, W. Tang and J. P. Collomosse, Eds., 173–180.

REINHARD, E., ASHIKHMIN, M., GOOCH, B., AND SHIRLEY, P. 2001. Color transfer between images. *IEEE Comput. Graph. Appl. 21*, 5 (Sept.), 34–41.

RODRÍGUEZ, A., GARCÍA, R. J., GARCÍA, J. M., MAGDICS, M., AND SBERT, M. 2013. Implementation of a videogame: Legends of girona. In *Actas del Primer Simposio Español de Entretenimiento Digital*, P. González Calero and M. Gómez Martín, Eds.

ROIG, G. 2010. *Peindre la lumière*. Guides Oskar.

SAUVAGET, C., AND BOYER, V. 2009. Abstraction of photographs: a comics style approach. In *Proceedings of the 3rd international conference on European computing conference*, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, ECC'09, 215–220.

SAUVAGET, C., AND BOYER, V. 2010. Stylization of lighting effects for images. In *Signal-Image Technology and Internet-Based Systems (SITIS), 2010 Sixth International Conference on*, 43–50.

SONG-HAI, Z., LI, X.-Y., HU, S.-M., AND MARTIN, R. R. 2011. Online video stream abstraction and stylization. *IEEE Transactions on Multimedia 18*, 6, 1286–1294.

SZIRMAY-KALOS, L., SZÉCSI, L., AND SBERT, M. 2008. *GPU-Based Techniques for Global Illumination Effects*. Synthesis Lectures on Computer Graphics and Animation. Morgan & Claypool Publishers.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *Proceedings of the Sixth International Conference on Computer Vision*, IEEE Computer Society, Washington, DC, USA, ICCV '98, 839–.

UNITY DEMO PROJECTS, 2013. Unity. http://unity3d.com/gallery/demos/demo-projects.

UNITY TECHNOLOGIES, 2013. Unity. http://unity3d.com.

VOLCKMAN, C., 2006. Renaissance. Producer: Onyx Films, France 2 Cinéma, Backup Films.

VRPLATFORM, 2013. http://recherche.parisdescartes.fr/Laboratoire MemoireCognition/Moyens-Techniques/VR-Platform.

WILLATS, J., AND DURAND, F. 2005. Defining pictorial style: Lessons from linguistics and computer graphics. *Axiomathes 15*, 319–351.

WINNEMÖLLER, H., OLSEN, S. C., AND GOOCH, B. 2006. Real-time video abstraction. *ACM Trans. Graph. 25*, 3 (July), 1221–1226.

WINNEMÖLLER, H. 2011. Xdog: advanced image stylization with extended difference-of-gaussians. In *NPAR*, ACM, J. P. Collomosse, P. Asente, and S. N. Spencer, Eds., 147–156.

YOUNG, R. A. 1987. The Gaussian derivative model for spatial vision: I. Retinal mechanisms. *Spatial Vision*, 273–293.

ZHAO, H., JIN, X., SHEN, J., AND WEI, F. 2009. Real-time photo style transfer. In *CAD/Graphics*, IEEE, 140–145.