1.

Hardware design and representation of graphics in videogames

A case study: the Sega Saturn

Marco Liboà

Transactions of the Digital Games Research Association
December 2019, Vol. 5 No. 1, pp. 1-43. ISSN 2328-9422

ABSTRACT

The paper focuses on how the design of the hardware supports and constrains the representation of graphics in videogames. The Sega Saturn was chosen as a platform of study due to the complexity of its internal circuitry and the period during which it was commercialised, characterised by a shift in the representation of game graphics from 2D to 3D. The peculiar characteristics of Saturn's two video display processors and the way they shape

the graphics of games developed for it are presented in a few selected examples. In particular, it illustrates how a 3D space can be simulated by means of 2D background layers, and how hardware limitations and different video-signals can affect the final rendering of game graphics. It concludes that different graphical techniques, present in a certain episode of a game series, could be absent in a direct sequel and then reappear all together in a later episode, leading to a non-linear technological innovation trajectory. Furthermore, it is ascertained that the Saturn hardware architecture influenced the efforts of developers in subtle and unexpected ways.

Keywords

Sega Saturn, Platform Studies, Hardware

INTRODUCTION

So far, videogames have been analysed from a variety of different perspectives, including, but not limited to, those of humanities, narratology, human-computer interaction, semiotics, discourse analysis, psychoanalysis, social, cultural and literary studies. All these different approaches have positively contributed to the development of the multidisciplinary field of game studies. In this article, I will focus on how the design of the hardware supports and constrains the representation of graphics in games developed for a specific platform, the Sega Saturn. In using the term platform, I mean computing platform, such as those that enable procedural works, i.e. works enacted by processes and algorithms executed by the platform itself (Montfort & Bogost, 2009).

The Sega Saturn is a Japanese videogame console, designed and produced by Sega. It was commercialised during the second half of the 1990s, a period characterised by the shift from 2D sprite-based to 3D polygon-based graphics. This shift is mirrored in the complex hardware of the system, designed to handle both

the graphics paradigm of the previous console generation and the new one that was starting to take hold in the arcades due to companies like Namco and Sega itself. In order to achieve such a feat, the Saturn relies heavily on parallel computation and is equipped with a dual physical CPU and two video display processors, the VDP1 and the VDP2, the former dedicated to the drawing of sprites and polygons, and the latter dedicated to the drawing of 2D background layers (Figure 1). Due to its hardware complexity, the Saturn is a notoriously difficult console to program; which, in my view, would be an interesting topic of study. Furthermore, as pointed out by Apperley and Parikka (2018), so far platform studies have focused on commercially successful consoles and home computers, and therefore, analysing a platform like the Saturn, which is considered a commercial failure, is valuable progress in further expanding the platform studies approach. Apperley and Parikka question the limitations of a platform study on a "failed" platform. Even though I do not consider the Saturn a failed platform, since it is still the most successful Sega console in Japan (Donovan, 2010), I nevertheless encountered some difficulties in creating a proper "archive". As explained by Apperley and Parikka, a platform studies' archive includes software developed for the platform, usually games, and "developer interviews, end user responses, and other material from the video game subculture that Mia Consalvo (2007) has dubbed "paratexts" – primarily journalism and marketing materials" (Apperley & Parikka, 2018). Although the archive that I used to study the platform is mostly comprised of games, I integrated it with the official development kit documentation found on internet forums – an indication of a past hobbyist dev scene, without which the documentation would have not been preserved – and with the use of emulators. Creating this last part of the archive for an unsuccessful platform is where I found most issues, compared to a successful one. Unfortunately, Saturn emulation is not in a very good state, presenting stability, fidelity and compatibility issues. For this reason, I also had to use an original Saturn console connected to a CRT TV to complete this case study.

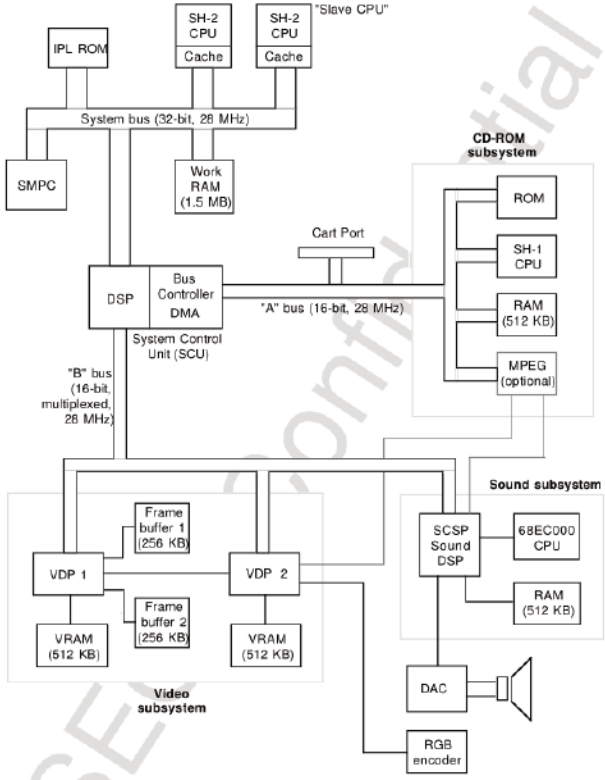

Figure 1: Sega Saturn block diagrams (Sega of America, 1994a)

Regarding the games selected as part of the platform archive, I will provide an analysis of a specific game series, *Virtua Fighter*, to show how the developers used 2D images to simulate a space that was 3D modelled in the original arcade version of the game, and developed for more computationally powerful hardware. Thereafter, I will illustrate how the hardware design of the console limits the usage of the half-transparent effect, a technique applied to represent translucent materials in graphics, and even how the type of cable used to connect the console to the TV can alter the final rendering of the image in a significant way.

A WORLD OF LAYERS

As pointed out by Arsenault in his book, "Super Power, Spoony Bards, and Silverware" (2017), on the Super Nintendo, when a new console is released, the available games need to fulfil a double purpose: they have to both persuade the gamers to buy the product, by offering established game genres, and attract developers to the new creative possibilities offered by the platform. The responsibility for showing the 3D capabilities of the Saturn fell upon the port of one of the most successful Sega arcade games at the time: *Virtua Fighter* (Sega 1993).

Virtua Fighter perfectly represents the characteristics of a launch title identified by Arsenault (ibid.): it is representative of a very well-established genre, the one-vs-one fighting game, and, at the same time, it completely revolutionises the graphical paradigm utilized in the genre by replacing 2D sprites and backgrounds with full 3D polygonal graphics. *Virtua Fighter* was one of the first 3D fighting games to appear in the arcades in the middle of the 1990s, and it was so successful that it even influenced the development of Sega's competitor console, the Sony PlayStation (Asakura, 2000). It was by no means the first 3D game: in fact, polygonal graphics had already been used in driving and racing simulators a few years before.1 However, *Virtua Fighter* was one of the first games to represent human characters by means of polygons. Even though it can be argued that, from a visual standpoint, *Virtua Fighter* characters do not look as detailed as a typical 2D character, the use of 3D graphics allowed developers to implement realistic animations and camera movement techniques that, back then, were considered innovative (Pettus, 2013).

Virtua Fighter was one of the most successful games in Japanese arcades, and Sega obviously ported it as a launch title for its main console. The Sega Saturn was released on 22 November, 1994, in Japan, and *Virtua Fighter* was one of the titles available at launch. The game was a commercial success, selling almost 1:1 with the console. From a marketing point of view, *Virtua Fighter* can be

considered a killer-application, i.e. a piece of software available exclusively for a specific platform, and that alone is a reason for the purchase of the new hardware. However, a very rushed development greatly hindered the quality of the porting, resulting in several graphical glitches, especially compared to the arcade version. For example, in the Saturn version, some of the polygons that shape the characters are affected by a flickering effect that is not present in the arcade version. The next paragraph, after a brief overview of the Saturn CPU and its video sub-system, describes how the original arcade version of the game was adapted to the hardware characteristics of the console.

In order to enable the Saturn to run ported arcade games that were developed on very powerful and expensive machines, Sega engineers relied heavily on parallel computation in designing its hardware architecture. Having the main CPU offload certain operations to other coprocessors was not, by any means, a novelty. For example, the Atari VCS was already equipped with a custom microchip, the *Television Interface Adaptor* (TIA) to handle the drawing of the image to the screen (Montfort & Bogost, 2009). In the Commodore Amiga, the Motorola 68000 CPU offloads some of its operations to three different microprocessors called *Denise, Paula* and *Agnus*. While Denise is responsible for drawing graphics, and Paula handles I/O and sound, Agnus manages both microprocessors to prevent conflicts with each other and the CPU (Maher, 2012). In a similar way, the Saturn CPU offloads graphical operations to the VDP1 and the VDP2, and the input and peripheral management to the *System Management and Peripheral Control* (SMPC), which is also in charge of coordinating the whole system, much like Agnus does in the Amiga. However, not only is the Saturn equipped with many microprocessors specifically designed for different tasks, it also has two physical CPUs, a pair of SH-2 Hitachi microprocessors, set in a master-slave configuration. One SH-2 is set as a master CPU, while the other is set as the slave, and is subordinate to the first CPU. Both CPUs have access to two megabytes of RAM through the system bus, and each has four kilobytes of cache memory. Since both CPUs

share the same memory bus, access to it is mutually exclusive, therefore, optimising the use of their internal cache memory is essential in order for the system to achieve maximum performance. However, as previously mentioned, the Saturn is also equipped with two video display processors, the VDP1 and VDP2, the former dedicated to drawing sprites and the latter to drawing background layers. The VDP1's capabilities include scaling, rotation and twisting of sprites, which are the modelling base of any 3D game on the Saturn, in a similar way as triangles are the basic element of 3D modelling nowadays.2 On the other hand, the main purpose of the VDP2 is to plot the "scroll screen" and it is usually employed to draw the background and the GUI in Saturn videogames. The scroll screen consists of different layers, defined either as screens or backgrounds in the official Sega documentation, each one entrusted with a specific set of operations and transformations. Specifically, the VDP2 can plot up to four different background layers that can be moved up, down, left or right. These are defined as normal backgrounds, from normal background zero (NBG0) to normal background three (NBG3). The NBG0 and the NBG1 can also be scaled horizontally or vertically (zoomed in or zoomed out). However, this feature is not available for the NBG2 or the NGB3, making the first two layers more suitable for drawing the actual background of a game scene, such as the sky, and the last two layers more convenient for drawing interface elements. Background layers that can also be rotated and scaled along their axes are defined as rotation background layers. The VDP2 can draw up to two rotation background layers at a time, with the identifiers being rotation background zero (RBG0) and rotation background one (RBG1). Finally, the last two screens that compose the scroll screen are the line colour screen and the back screen. The line colour screen is a special layer used for colour calculations, while the back screen is the default colour to be displayed for pixels that are not covered by any other layer. The VDP2 can, at the same time, plot up to either four normal backgrounds, or three normal backgrounds and one rotation background, or only two rotation backgrounds. By carefully designing and combining all the different background

layers, developers can simulate a three-dimensional space by using only two-dimensional planes. For example, a common technique is parallax scrolling, i.e. “the movement of different background layers at different speeds to simulate a depth of field that increases the perceptual illusion of perspective” (Arsenault, Côté, Larochelle & Lebel, 2013). Another common technique to simulate a three-dimensional space with a planar projection is the usage of a rotation background as the surface over which the action takes place. This technique is extensively used in videogames made for the Super Nintendo Entertainment System (ibid.) and it is possible to reproduce it on the Saturn by using a rotation background.

Figure 2a: Virtua Fighter on Sega Saturn

Figure 2b: VDP1 framebuffer (sprites plane)

Figure 2c: NBG0 (interface)

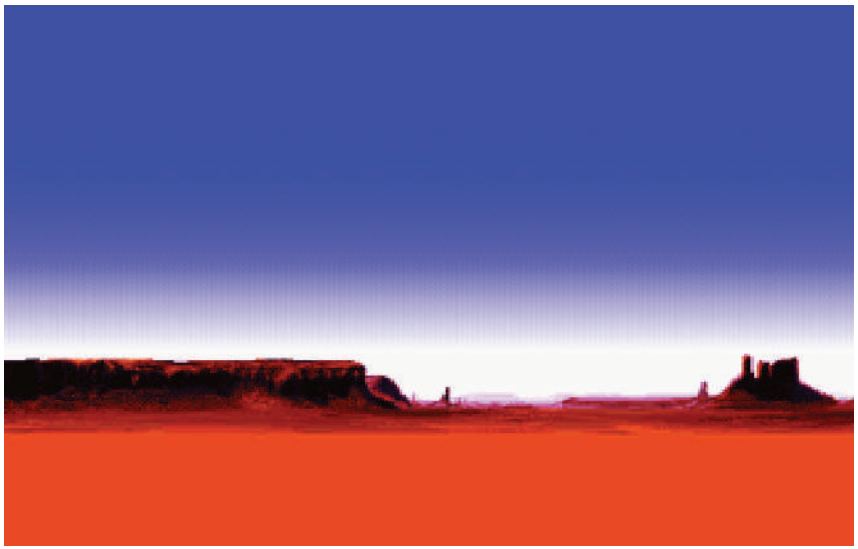

Figure 2d: NBG1

Unsurprisingly, in the Saturn porting of *Virtua Fighter*, the VDP1 is used to draw the two characters and the ring, while the VDP2 is used to draw the background and the interface. A standard scene from the game is composed of the two fighters in a ring, a distant background representing the stage, and the game interface (Figure 2a). The information shown in the interface at the top of the screen includes the health bars and the name of the characters, along with the round countdown timer and the victory points. At the bottom of the screen, a number indicating the round, a pause message and the total play time are displayed. In order to help the reader understand which elements in the scene are drawn by each video processor, a series of screenshots, with the different screen layers turned on and off, will be used throughout this paper. As shown in Figure 3, the game interface is mostly drawn by the VDP2 in NBG0, with the notable exceptions of the victory points and the total play time in the bottom right-hand corner, which are both drawn using sprites. It is worth noticing that, while the round countdown timer is part of the interface, the total play time is drawn by using sprites. One possible reason could be that the interface is reset at the start of each round (both health bars and timer) while the total play time

has to be stored between rounds. If the total play time had also been part of the interface, it would have been reset as well, losing the information it was supposed to show.

Figure 3: NBG0 disabled, part of the interface is no longer visible.

Each stage of the game is represented by a different bitmap image. At the start of the round, the bitmap is loaded in the portion of memory reserved for the NBG1. Once the fight starts, and during the replays, the bitmap is moved along the horizontal and vertical axis according to the movement of the camera, so as to give the appearance of a 3D space around the fighters. However, since the distant background is just a plain image that cannot dynamically adjust its linear projection based on the point of view of the camera, the bitmap itself has to be drawn in a specific way. Basically, the bottom part of the bitmap is used to simulate the presence of a terrain around the ring (Figure 2d). In fact, in the scene there is an invisible plane, parallel to the ground, which is used as a collider on which the characters land when they are thrown out of the ring. The part that represents the terrain is just a vertical background, mono-coloured in order to look identical from any point of view. Even when the camera frames the scene from up above during a replay, the background is drawn in such

a way that only the single coloured part of the NBG1 is actually visible. The representation of a 3D space is achieved by drawing both the ring and the characters using polygons, accordingly to linear projection rules. Furthermore, the use of polygons allowed developers to simulate the presence of a landscape that is very far away by zooming in and out of the characters and the ring when necessary, while keeping the background image fixed, in order to add depth to the scene.

A few months after the release of *Virtua Fighter* for the Saturn, Sega released an updated version titled *Virtua Fighter Remix* (Sega 1995), which was given for free to all Saturn users in the U.S. as an apology for the poor porting of the previous game (Pettus, 2013). *Virtua Fighter Remix* is an improved version of the original game with characters made of textured polygons instead of flat shaded polygons. The next paragraph shows how developers assigned the computation of the graphical elements of the game to the various Saturn microchips, in order to deliver a better-looking game just a few months after the release of the original port.

Figure 4: Texture mapped characters in Virtua Fighters Remix.

Figure 4 shows the different appearance of the characters, which are drawn using textured polygons instead of flat shaded polygons. Taking into consideration how the platform already struggled to port the original game, it would be interesting to investigate how the developers managed to find new computational resources to use texture mapping, instead of flat shading. An analysis of the NBG0 and the NBG1 reveals that these two layers are still used to draw, respectively, the interface elements and the distant background, as shown in Figures 5b and 5c. Instead, an investigation of the RBG0 reveals that in *Virtua Fighters Remix* the ring is not drawn using sprites, but its surface is a rotated plane, as shown in Figure 5d. The surface of the ring is a single bitmap image rotated and scaled to simulate a flat horizontal plane, an effect similar to Super Nintendo's "mode 7" (Arsenault & Côté, 2013). By using this technique, the developers were able to reduce the number of polygons drawn on the screen, freeing some resources from the VDP1 and increasing the computational load on the VDP2. These resources could then be used to draw more detailed characters. It is important to remember that each of the Saturn video graphic processors has its own bank of VRAM: not using the VDP2 to its fullest means wasting some precious memory, since the VDP1 cannot access the VDP2 memory. As a last detail, the sides and the edge of the ring are still drawn using polygons (figure 5a). However, it seems that texture mapping hindered programmers from using dynamic lighting in the game. This is especially visible in stage 3: in the original *Virtua Fighter* port, the floor is a source of light that illuminates the characters from below (Figure 6a); in *Virtua Fighter Remix* there is no light source and no shadows are drawn on the characters. Self-shadowing is, in fact, absent in all stages of this second version of the game (Figure 6b). This is an interesting example of how, due to the platform limitations, a technological innovation present in *Virtua Fighter*, like dynamic lighting, had to be removed in its remake in order to use another technological innovation: texture mapping. One would expect the trajectory of a technological innovation (Arsenault & Côté, 2013) to always move forward, but instead, in the case of the Saturn, it seems to have followed

a fluctuating curve. As a matter of fact, texture mapping and dynamic lighting were used at the same time on the Saturn only in the second half of its life-cycle (Figure 7). Not even the much-praised Saturn port of *Virtua Fighter 2* (Sega 1994) achieved this feat, mostly because it runs in high resolution and, in this mode, the lighting calculation is disabled due to a hardware limitation.3 Despite this restriction, the Saturn version of *Virtua Fighter 2* is one of the most important examples of how improved development environments and tools can enhance the game making process. *Virtua Fighter 2* is one of the first games developed with the Saturn Graphics Library (SGL), a library written in C language and released by Sega in order to ease the development of 3D games on Saturn. From a technical perspective the game is considered a masterpiece, running in high resolution, double density interlaced mode (704*448) at 60 frames-per-second and simultaneously using four background layers: NBG0, NBG1, NBG2 and RBG0. However, being a port of a game originally developed for much more computationally powerful hardware − the Sega Model 2 − the developers had to compromise on the final graphic fidelity. Characters, models, textures and animations are very similar to the arcade counterparts, apart from some minor differences (e.g. the hair animation of most characters was removed). As previously mentioned, the biggest difference regarding the characters is the absence of dynamic lighting in the scene and, consequently, in the Saturn version there are no shadows on the 3D models of the fighters, as seen in Figures 8a and 8b. The two most important technological innovations introduced by *Virtua Fighter 2* (especially in comparison with the original *Virtua Fighter)*4 were the adoption of texture mapping for the characters and the presence of 3D-modelled structures in the space around the ring. Taking into consideration the computational power difference between the Saturn and the Model 2, it was not possible for the developers to fully model the background in 3D on the console version, so they had to rely on the peculiar 2D capabilities of the VDP2 in order to provide a port as faithful as possible to the original arcade game. In *Virtua Fighter 2*, the interface elements are drawn in the NBG2, as demonstrated in Figure 9e. The NBG0 displays a 2D image

that represents the 3D background elements of the arcade version, while the NBG1 is used to draw the distant background as seen in *Virtua Fighter* (Figures 9c and 9d). As previously mentioned, the ring in *Virtua Fighter Remix* is drawn using the RBG0, however, in *Virtua Fighter 2*, sprites are only used to draw the sides of the ring and not the edge (Figures 9f and 9b). By scaling and moving the NBG0 and the NBG1 at different speeds with respect to one another, the developers managed to create a parallax effect that simulates the presence of an actual 3D space between the ring and the first background image (NBG0), and between the latter and the second background image (NBG1).

Figure 5a: VDP1framebuffer

Figure 5b: NBG0

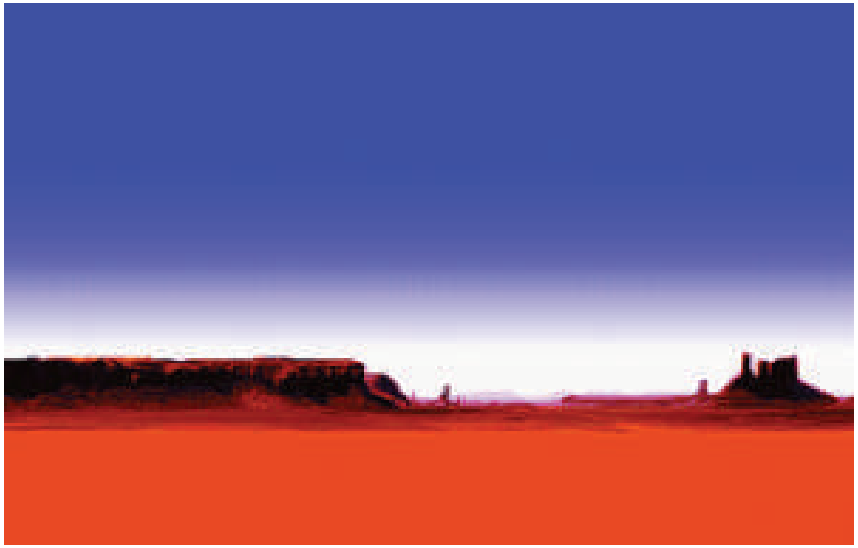

Figure 5c: NBG1

Figure 5d: RBG0 (the ring)

Figure 6: Lightning source and shadows

Figure 6b: Missing shadows on characters

Figure 7: Texture mapping and dynamic lightning in Fighters Megamix (Sega 1996)

However, this technique has some trade-offs, and it was not possible to reproduce every stage of the arcade version faithfully. For example, in the first stage of the game a road is present in the

background. In the console version, the lines that form the road are not dynamically updated according to the rules of perspective when the camera moves, while in the arcade version they are correctly distorted to create a vanishing point (Figures 10a and 10b). The third stage was also heavily reworked in the Saturn version. In the arcade version of the game, this stage was designed as a graphical showcase, with the two opponents fighting on a raft floating on a river passing under huge, fully 3D-modelled bridges, that project their shadows on the fighters (Figure 11a). Since it was not possible to recreate the bridges with 2D elements, the entire stage had to be redesigned. The Saturn version of stage 3 is very similar to all other stages, with a static ring placed on the bank of a river (Figure 11b).

Figure 8a: Shadows in Virtua Fighter 2 on the Model2 arcade version.

Figure 8b: Shadows are not present in the Saturn porting of Virtua Fighter 2

Figure 9a: Virtua Fighter 2 on Sega Saturn

Figure9b: VDP1 framebuffer

Figure 9c: NBG0

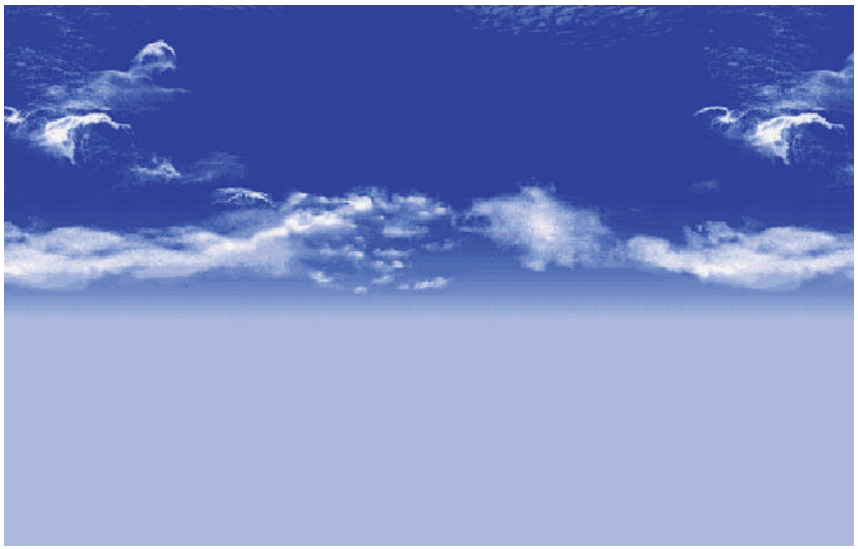

Figure9d: NBG1

Figure 9e: NBG2

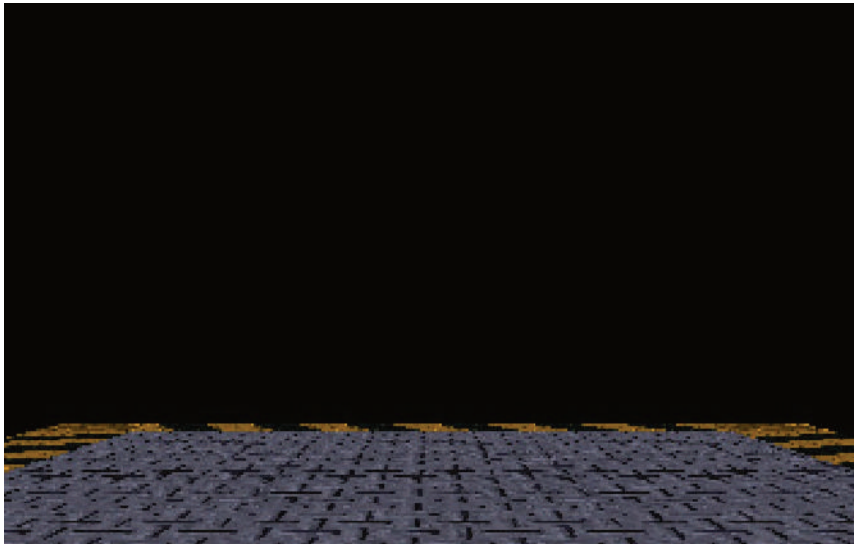

Figure 9f: RBG0

Figure 10a: Virtua Fighter 2, Model 2 version (SEGA Model 2 Emulator v.1.1a, ElSemi, 2014). The road in the background has a vanishing point and is drawn accordingly.

Figure 10b: Virtua Fighter 2, Saturn version. The road is just shifted to the left and is not drawn according to linear projection rules.

Figure 11a: Virtua Fighter 2, Model 2 version (SEGA Model 2 Emulator v.1.1a, ElSemi, 2014). The bridge is 3D modeled and shadows the ring.

Figure 11b: Virtua Fighter 2, Saturn version. The bridge is not present in stage 3, however the water of the river is animated.

In conclusion, being a machine designed to handle 2D graphics, the Saturn's developers struggled at first to reproduce 3D graphics. The proper usage of the VDP2 and the several background layers that were available to the Saturn were the keys to achieve the maximum performance from the system. It was especially important to design 2D elements in a way that could give the illusion of a 3D space. Furthermore, the release of the Saturn Graphics Library allowed programmers to more easily access all of the Saturn's hardware resources, and the more faithful port of *Virtua Fighter 2* compared to the first release of the series stands as proof of this improvement.

Figure 12: Mesh effect in Saturn games left, Nights Into Dreams… (Sega 1996); right, Thunder Force V (Technosoft 1997)

THE HALF-TRANSPARENT EFFECT

In computer graphics, half-transparency is a technique used to blend the colours of two overlapping pixels so that one of them appears to be transparent or translucent. This technique is widely used to simulate water, flames and smoke in games. One of the most controversial aspects of the Saturn hardware architecture lies in whether or not it was capable of drawing half-transparent pixels on the screen. This subject was one of the most debated during the console's life-cycle, and, despite it not being the object of this research, I think it is important to shed some light on this aspect, at least from a technical point of view. At the centre of the debate was the highly controversial "checkered" effect that is used in most Saturn games to provide computationally lightweight half-transparency (Figure 12). According to Sega's official documentation, a "mesh"5 is a sprite in which only every alternate diagonal of pixels is drawn (Figure 13). Mesh sprites show a typical checkered pattern in which, given a scanline, either odd or even pixels are drawn, while the rest are left fully transparent (invisible or completely see-through). Even though, both from a programming and a hardware resource point of view, using a mesh is a cheap and easy way to deliver a semi-transparent effect, its visual quality cannot match a standard half-transparency, in which the colour of every overlapping pixel is averaged with the background. Sega's competitor consoles6 were considered superior in this regard, since they were able to draw half-

transparent polygons in a consistent way, without the need to rely on visual tricks. As a matter of fact, it is actually possible to produce a regular half-transparent effect even on the Saturn, as shown in Figure 14. Nevertheless, despite the Saturn being able to draw half-transparencies, many games still rely on the checkered effect. In order to discuss the reason behind this practice, I need to briefly explain some specific features of the Saturn hardware architecture and its 2D/3D dual nature.

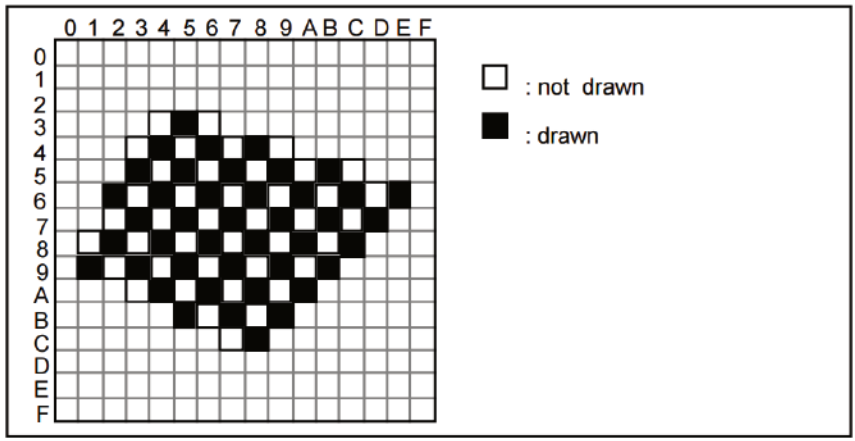

Figure 13: Mesh pattern (Sega of America, 1994c)

Figure 14: The water is half-transparent and the monster can be seen through its surface Panzer Dragoon 2 Zwei (Sega 1996).

According to Sega's official documentation, the VDP1 is in charge of the half-transparencies of sprites placed on top of other sprites, while the VDP2 has to deal with the half-transparencies between sprites and backgrounds. However, the VDP1 can handle the half-transparent effect only on pixels stored in its framebuffer in RGB

colour mode, since it needs the actual colour information to perform the requested calculation, i.e. a colour average between the pixel already drawn in the framebuffer and the pixel that is about to be plotted. Despite the VDP2 being technically able to perform the half-transparency, regardless of the colour mode in use, this is not feasible in practice. The VDP2 does not store any sprite information, since it only deals with background layers, and, indeed, it sees the VDP1's framebuffer simply as an extra layer on top of the background images stored in its VRAM7. When a pixel descriptor, i.e. the block of memory in the framebuffer that contains the colour information of a single pixel, is set to RGB colour mode, the VDP2 can only make the entire framebuffer, meaning every single sprite, half-transparent with respect to the other background layers. This hardware constraint really limits the use of the RGB colour mode since it is unlikely to find a situation in a game where each and every sprite is rendered semi-transparent. However, RGB is not the only colour mode available to programmers of this console. Sprites can also be drawn using the palette colour mode, in which case, three out of the 16 bits of the pixel descriptor are reserved for a priority code8 and the VDP2 can be set to render half-transparent only the sprites with a priority code above or below a certain threshold. However, unfortunately, the VDP1 cannot calculate half-transparency on sprites in palette colour mode. To recap: a sprite that has to be made half-transparent on top of other sprites should be drawn in RGB colour mode with the VDP1 handling the colour calculation; a sprite that has to be rendered half-transparent on top of other background layers should instead be drawn in palette colour mode with the VDP2 handling the colour calculation. This hardware design severely limits the use of half-transparent sprites drawn on top of either sprites or background elements, and is the main reason for the widespread use of the checkered effect in Saturn games, which is an easy and computationally faster way to draw them on both.

Figure 15: Mesh effect on a CRT TV connected through an RGB SCART cable (top) and a composite cable (bottom).

Figure 16a: The girl's cloak is half-transparent compared to the background.

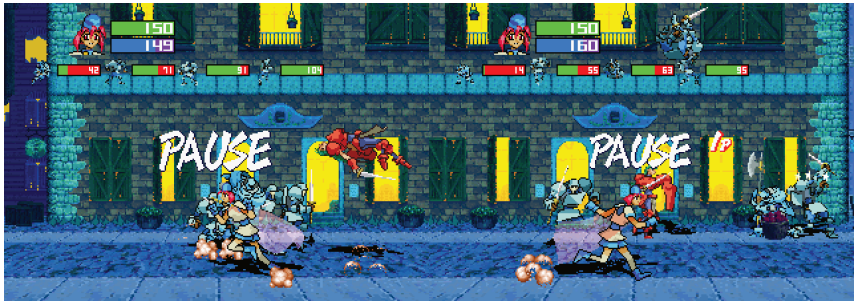

Figure 16b: Sprites drawn behind the girl's cloak are not visible.

Furthermore, if a composite video cable, like the one included as the standard cable in the console package, is used to connect the console to a TV, the visual end result is quite similar to a true half-transparency, since an image broadcast using a composite video cable is never crisp and the pixels tend to mix their colour with adjacent ones9. Due to the analogue nature of the signal and the modulation technique used to compress the signal in one single channel10 (unlike the two channels of s-video and three channels of RGB), when the image is reconstructed on a CRT TV, it looks somewhat blurry, meaning that the colour information of a pixel is averaged with the colour information of other pixels in its proximity. The mesh effect takes advantage of the low quality of the video signal to average the colour of its checkered pattern with the background, resulting in an effect that looks very similar to half-transparency (Figure 15). It is indeed fascinating how game developers managed to use the characteristics of cables, video

signals and CRT TVs to obtain a computationally complex graphical effect, without the hardware doing any calculation at all. It could be argued that even the cable and CRT TV take part in the processing of the video output of the game console, thus extending the concept of platform way beyond the chassis of the console itself. The next paragraph illustrates some examples of typical visual artefacts present in Saturn games that make use of the half-transparent effect.

In *Guardian Heroes* (Treasure 1996) there is an example of the VDP2 half-transparent effect between a sprite and a background image. As seen in Figure 16a, the cloak of one of the game's characters is rendered half-transparent, and it is possible to see the background through it. Since a half-transparency between a background and a sprite is handled by the VDP2 and not every single sprite is made half-transparent, it is possible to infer that the cloak is rendered using a palette colour. However, when the girl's cloak is drawn on top of another sprite, since the VDP1 can handle colour calculation only for sprites in RGB colour mode, the cloak covers the sprite behind it, despite being half-transparent (Figure 16b). In this case, the developers opted to use a half-transparency instead of a mesh effect, even though the cloak could have been drawn on top of both sprites and backgrounds. It could be argued that they deemed the visual glitch caused by the overlapping of the cloak with other sprites of minor importance compared to the higher overall graphics quality of the effect. In fact, with the very fast action of the game, and with many sprites on the screen at the same time, it is rather difficult to notice the glitch during a normal play session. In the next paragraph, I will analyse another game as a further example of the limitations of the half-transparent effect of the Saturn.

Figure 17a: Objects near the horizon are drawn half-transparent with a gradient based on the distance from the camera (fade-in effect).

Figure 17b: The barrier is a VDP1 half-transparent effect.

Figure 17c: The ground (RBG0) is not visible behind the barrier, despite it being half-transparent (right: RBG0 disabled).

Sonic R (Sega, 1997) is arguably one of the most graphically advanced games ever made for the Saturn. It is most famous for its fade-in effect, used to cover the pop-up of objects far away from the camera (Figure 17a). This effect is created by the capability of the VDP2 to use different blending ratios for half-transparency, enabling a sprite to gradually blend with a background. When the object reaches a certain distance from the camera, it is finally made fully opaque. However, Sonic R also contains a few examples of half-transparencies created by the VDP1, most notably the barrier power-up effect (Figure 17b). As expected, the simultaneous use of the half-transparent effect by both the VDP1 and the VDP2 can cause some visual artefacts. For example, in Figure 17c, it is possible to see how the barrier is half-transparent only when sprites, which are actually drawn by the VDP1, are behind it, and how it covers elements from the rotation background. It is possible to speculate that this is one of the main reasons why, despite the programmers demonstrated knowledge of the Saturn hardware, they still decided to use the mesh effect in order to draw the characters' shadows. Since the characters can run on parts of the track that could be made by either polygons or a rotated background, if the shadows had been drawn using the VDP1 half-transparent effect, they would have covered part of the floor when drawn on the rotation background, resulting in a fully opaque black circle. The fact that the barrier effect could have caused graphical glitches, with half-transparent polygons rendered using the VDP2, is even more evident in the last track of the game,

Radiant Emerald. In this track the fade-in effect is disabled (and the polygon pop-up effect is clearly visible) since the entire track is made permanently half-transparent to convey the idea of running on some sort of crystal material (Figure 18). Most notably, in this track there are no power-ups available, so it is not possible for the player to obtain any type of barrier. If that had been the case, the barrier would have hidden part of the track behind it. Therefore, the developers had to drop a gameplay element, the barrier power-up, in favour of an aesthetic element: rendering the entire track half-transparent.

Figure 18: In the Radiant Emerald the track is half-transparent (left). The fade-in effect is disabled (right)

This article gave a concise overview of how the half-transparent effect is implemented in the Sega Saturn console. It briefly explained how the Saturn's two custom video processors, the VDP1 and the VDP2, manage this effect, and what type of limitations and interactions they have when used together. Then, the article provided a few practical examples of half-transparencies in a few selected games, in order to give the reader a better idea of what type of visual constraints programmers, game designers and artists alike had to work with when creating games for the Sega console. It is important to point out how the Saturn hardware architecture forced programmers to rely on the mesh effect for half-transparency, both for technical and economic reasons (when a sprite has to be drawn on top of both sprites and backgrounds, the checkered effect is the fastest and cheapest method), since it is indeed possible to program true half-transparencies on the Saturn, but it requires a mastery of the hardware that not many

programmers had in the early years of the console's life-cycle. Investing resources to attain such knowledge was expensive, and it was probably not deemed viable by the investors, considering the console's limited sales figures.

CONCLUSION

In my analysis of the first three *Virtua Fighter* games, I illustrated how the hardware could limit and define the approach that designers, artists and programmers had to take in order to represent a 3D space in Saturn games. Since the Saturn could not faithfully reproduce the graphics of the original arcade games, programmers and artists had to rely on the parallax effect and on the rotation and scaling of backgrounds to visually simulate a 3D space around the game's characters. Another element that came to light is how programmers had to forgo dynamic lighting, adopted in *Virtua Fighter*, in favour of texture mapping, when porting *Virtua Fighter Remix* and *Virtua Fighter 2*. As previously pointed out, it could be expected that the trajectory of a technological innovation would always move up and forward, but instead, with regard to the Saturn, two different directions have been taken: dynamic lighting and texture mapping. Eventually, those two techniques were finally used together in the second half of the console's life-cycle, when programmers mastered the hardware and were able to simultaneously use both techniques in the same game.

Another typical aspect of the Saturn hardware is that the game's graphics do not always appear as the developers intended. For example, it is not enough for a developer to simply set a sprite to be half-transparent: depending on the colour mode and the position of the sprite (on top of another sprite or a background), the final rendering could be very different, ranging from a proper half-transparency to a part of the image mysteriously disappearing. Also, the type of cable used to connect the platform to the TV contributes to the final image rendering, sometimes even with a positive outcome, as is the case of the half-transparency created by

the mesh effect when using a composite video cable and a CRT TV.

Finally, the Saturn hardware can, in particular occurrences, limit the game design space in unexpected ways, as seen in *Sonic R*: due to a hardware limitation, power-ups could not be placed in the last track of the game in order to avoid a graphical glitch. However, it could be argued that platform limitations can be an incentive for developers to be creative, challenging them to overcome such constraints in order to maximise the capability of a given platform.

APPENDIX: SEGA SATURN BIBLIOGRAPHIC DESCRIPTIONS

In I AM ERROR, Altice points out how bibliographic descriptions are an underrated subject in game studies, whether it be in books or articles. Although addressing the lack of well-defined norms for citing videogames in literature is not within the scope of his work, in his book he introduces two different bibliographic formats for videogame sources: one for physical objects (e.g. cartridges, disks) and another for files used in an emulated context (e.g. ROMs, save files). Furthermore, he also explains how "rich bibliographic records necessarily require a baseline technical understanding of the objects they describe" (Altice, 2015, p. 336), meaning that choosing which fields are used to describe the object under investigation is both strictly connected to the technical features of the object itself and to the writers' understanding of such technical features. Consequently, a bibliographic format for a specific platform might not be suitable to describe a videogame designed for a different one. Altice rightly points out how a "bibliographic description that suits a Famicom cartridge will not necessarily suit a ColecoVision cartridge" (ibid.). In light of this consideration, for this paper, I am going to use two modified versions of the bibliographic formats used by Altice for the NES platform; one for physical sources and one for emulated sources. Such formats were

modified to take into account the specific technical features of the Saturn as a platform.

Format 1: Enumerative type for citing Saturn-compatible CD-ROMs and cartridges. *Title*. TV format [Region], Catalog ID, Media [Disc size]. Developer {Credits}: Publisher, Release date.

Format 2: Enumerative type for citing Saturn-compatible disc images/patches/save states used in emulation.

Original CD-ROM title [Type]. Author. "Filename and extension" (File size). Image type {audio subchannels file}. Date modified. Emulator [Virtual Disc Drive] {Optional cartridge}, BIOS version. <Download source>

The Catalog ID is an alphanumeric string usually present on the CD cover and sometimes also in the booklet or on the CD case, and is used to identify the revision of the game. The Virtual Disc Drive is an optional field to specify which software was used to emulate the CD-ROM disc drive, in case the emulator used requires it to read CD-ROM images. Finally, an emulator might require a BIOS file to run, so another field was added to specify which version is used. The above formats are just suggestions of what a possible model for a Saturn bibliographic description could be like. Since a bibliographic record depends on the technical understanding of the object of study, the above formats are very likely to be expanded and revised as the knowledge of the object under scrutiny deepens.

ENDNOTES

1. For example, the previous year Sega released its first 3D racing simulator: *Virtua Racing*.
2. In this paper, I consider the term sprite and polygon as equivalent.
3. In order to use the colour calculation feature of the VDP1 (Gouraud shading, half-luminance, half-

shadowing, etc.) the information regarding each pixel in the framebuffer is saved as a 16-bit block of memory and the most significant bit (MSB) specifies whether the colour calculation is on (MSB=1) or off (MSB=0). When the screen resolution is set to high definition, the information per pixel in the framebuffer is reduced to 8 bits to store twice as many pixels (only the horizontal resolution is doubled in high definition mode). However, each bit is used to represent a colour code and the information regarding the colour calculation is lost.

4. It is worth noting that *Virtua Fighter Remix* was ported to the Sega ST-V arcade board after the release of *Virtua Fighter 2*.
5. Nowadays, in computer graphics, a "mesh" is a solid made of polygons.
6. The Sony PlayStation and the Nintendo 64.
7. It is important to point out that the VDP2 is in charge of composing the final image (framebuffer and background layers), and sends it to the RGB encoder.
8. The priority code is used, among other things, to determine which sprites are in front or behind each background layer.
9. This image artefact is called pixel bleeding effect.
10. The luminance signal (black and white values) and the chrominance (saturation and hue information) travel on the same channel, using a frequency-division modulation.

BIBLIOGRAPHY

Altice, N. *I am error: The Nintendo family computer/ entertainment system platform.* Platform studies. Cambridge, MA: MIT Press, 2015.

Antime. (2002). Sega Saturn Official Documentation. Retrieved from http://koti.kapsi.fi/~antime/sega/docs.html

Apperley, T., and Parikka, J. “Platform Studies’ Epistemic Threshold.” In *Games and Culture* vol. 13, no. 4(2018): 349–69.

Arsenault, D. *Super Power, Spoony Bards, and Silverware. The Super Nintendo Entertainment System.* Platform studies. Cambridge, MA: MIT Press, 2017.

Arsenault, D., and Côté, P. “Reverse-engineering graphical innovation: an introduction to graphical regimes.” In *GAME The Italian Journal of Game Studies,* vol. 1, no. 2(2013). http://www.gamejournal.it/reverse-engineering-graphical-innovation-an-introduction-to-graphical-regimes/

Arsenault, D., Côté, P., Larochelle, A., and Lebel, S. “Graphical technologies, innovation and aesthetics in the video game industry: a case study of the shift from 2d to 3d graphics in the 1990s.” In *GAME The Italian Journal of Game Studies,* vol. 1, no. 2(2013). http://www.gamejournal.it/graphical- technologies-innovation-and-aesthetics-in-the-video-game-industry-a-case-study-of-the-shift-from-2d-to-3d-graphics-in-the-1990s/

Asakura, R. *Revolutionaries at Sony: The making of the Sony PlayStation and the visionaries who conquered the world of video games*. New York: McGraw-Hill, 2000.

Bogost, I., and Montfort, N. “Platform Studies: Frequently Questioned Answers.” *UC Irvine: Digital Arts and Culture 2009*. Retrieved from https://escholarship.org/uc/item/01r0k9br

Consalvo, M. *Cheating: Gaining advantage in videogames.* Cambridge, MA: MIT Press, 2007.

Donovan, T. *Replay: The history of video games*. East Sussex, England: Yellow Ant, 2010.

International Communication Union. (2007). Characteristics of composite video signals for conventional analogue television systems. Retrieved from https://www.itu.int/rec/R-REC-BT.1700-0-200502-I/en

Lowscore Boy. (2015). Sega Saturn Graphic In-depth Investigations. Retrieved from https://www.youtube.com/watch?v=f_OchOV_WDg

Maher, J. *The Future Was Here. The Commodore Amiga*. Platform Studies. Cambridge, MA: MIT Press, 2012.

Montfort, N., and Bogost, I. *Racing the beam: The Atari video computer system*.

Platform studies. Cambridge, MA: MIT Press, 2009.

Pettus, S. *Service games: The rise and fall of Sega* (Enhanced edition). [United States], Lexington, KY, 2013 Sega of America. (1994a). Introduction to Saturn Game Development. Retrieved from http://koti.kapsi.fi/~antime/sega/files/13-APR-94.pdf

Sega of America. (1994b). Saturn Overview Manual (temporary version 1). Retrieved from http://koti.kapsi.fi/~antime/sega/files/ST-103-R1-040194.pdf

Sega of America. (1994c). VDP1 User's Manual. Retrieved from http://koti.kapsi.fi/~antime/sega/files/ST-013-R3-061694.pdf

Sega of America. (1994d). VDP2 User's Manual Version 1.1. Retrieved from http://koti.kapsi.fi/~antime/sega/files/ST-058-R2-060194.pdf

Sega of America. (1995). Technical Bulletins. Retrieved from http://koti.kapsi.fi/~antime/sega/files/Sattechs.pdf

Sega Retro. (2006). Sega Saturn. Retrieved from http://segaretro.org/index.php?title=Sega_Saturn&oldid=321310

Sega-16. (2012). Inconsistent transparent effects on Saturn games. Retrieved from http://www.sega-16.com/forum/showthread.php?19962-Inconsistent-transparent-effects-on-Saturn-games

Shima. SSF emulator. Retrieved from http://www.geocities.jp/mj3kj8o5/ssf/index.html

Yabause Team. Yabause. Retrieved from https://yabause.org/

Sega Saturn CD-ROMs and disc images

Guardian Heroes. NTSC [JP], GS-9031, CD-ROM [531 MB], Treasure: Sega Of Japan, January 1996.

Nights Into Dreams… . NTSC [US], 81048, CD-ROM [524 MB], Sonic Team: Sega Of America, August 1996.

Panzer Dragoon 2 Zwei. NTSC [JP], GS-9049, CD-ROM [453 MB], Team Andromeda: Sega Of Japan, March 1996.

Sonic R. NTSC [US], 81800, CD-ROM [622 MB], Travellers Tales: Sega Of America, November 1997.

Thunder Force V. NTSC [JP], T-1811G, CD-ROM [523 MB], Technosoft: Technosoft, July 1997.

Virtua Fighter [Disc image]. "330 Virtua Fighter (U).mdf" (567 MB). MDF. 24/12/1996. SSF 012_beta_R3 [DAEMON Tools Lite 10.3], BIOS v1.00a (US, 1995).

Virtua Fighter Remix [Disc image]. "333 Virtua Fighter Remix (U).mdf" (571 MB). MDF. 24/12/1996. SSF 012_beta_R3 [DAEMON Tools Lite 10.3], BIOS v1.00a (US, 1995).

Virtua Fighter 2. NTSC [US], 81014, CD-ROM [627 MB], Sega AM2: Sega Of America, December 1995.