An Innovative Methodology for Hybrid Rendering at Interactive Rates Combining Screen Space Reflections with GPU Ray Tracing

Daniel Valente de Macedo, Maria Andréia Formico Rodrigues (Orientadora)
Programa de Pós-Graduação em Informática Aplicada (PPGIA)
Universidade de Fortaleza (UNIFOR)
Av. Washington Soares 1321, Bloco J, sala 30 – Fortaleza – CE – Brazil
{*danielvalentemacedo,andreia.formico*}*@gmail.com*

***Abstract*—The realistic representation of light within a computational domain is crucial in computer graphics applications (from the film industry to the computer games), however, it is not a trivial task and an extremely costly performance wise, in most cases, unfeasible to be performed in real time. Traditionally, realistic static scenes have been generated using ray tracing. For dynamic scenes, rendering algorithms at the GPU level have prioritized speed over quality to achieve interactive frame rates, with a drawback of generating undesirable visual artifacts. This work presents an innovative methodology for rendering in a hybrid way realistic scenes (both static and dynamic) at interactive rates. To generate reflections, it combines Screen Space Reflection (SSR) with ray tracing in GPU and, for shadows, it uses shadow maps and/or ray tracing. Systematic tests using varied scenarios (with flat, curved, rigid, and deformable objects) showed that this innovative methodology is able to generate high quality reflections at interactive rates (on average of 30 frames per second) in dynamic scenes, including multiple recursive rays.**

***Keywords*-Realistic Reflections; Dynamic Scenes; Real-time; 3D Scenes; Raytracing**

I. Introduction

In recent years, 3D graphics applications have been increasingly present in many relevant areas: entertainment, commercials, multidimensional data visualization and exploration, simulation on engineering and medical processes, etc. In particular, the level of quality required by the users of these applications has increased, aiming for very realistic, accurate and fast results. Consequently, this high demand has stimulated the emergence of faster graphics cards and more robust processors with high parallel processing capacity. However, even with the evolution of these hardwares, the current graphics cards still have important limitations, both in terms of quality and efficiency of the images generated.

Computer games, movies, and visualization of large three-dimensional data sets has a high dependency on the rendering process and target very important aspects of the computer graphics, such as: processing time, visual realism, interactive frame rates, robustness in terms of scalability (number of polygons processed over time and update rates on geometries of moving objects), resolution of generated images, etc. The GPUs are essential parts of graphics cards which influence directly the performance of the application. Recently, due to the high demand on parallelism provided by the GPUs and to take advantage of its robustness, the costly and traditional algorithms based on ray tracing began to be exploited in GPU, focusing on the rendering of 3D scenes.

Initially, visual effects (like shadows, reflections, indirect light, etc.) in interactive scenes were created using precomputed texture maps. As a result, the first examples of very simple 3D static scenes containing a quite limited number of rigid moving objects with low geometric complexity were produced, synthesized by ray tracing [1]. Currently, the ray tracing technique is not yet a widespread feature in interactive graphics applications, even in the area of digital games. In general, this is due to its high computational cost, which ends up compromising the user experience by not yet achieving interactive rates (FPS) with the current hardware. In particular, shadows and reflections are relevant effects, which add value to the generated images and, consequently, to the user experience, achieving a satisfactory quality result, when perceived by the human eye, but still rather complex to be synthesized by computer in real time.

Very recently, few hybrid solutions combining rasterization and ray tracing have been proposed. The one that most relates to this work is from Ganestam and Doggett in GPU [2], in which ray tracing is used to compute reflections of objects that are close to the observer, and a cubemap of g-buffers is used to render objects which are located further away from the observer.

There are two fundamental differences between the Ganestam and Doggett's solution [2] and the one presented in this work: (1) in this work, reflections are always provided, whereas in the work proposed by the above cited authors, the reflections can fail when an object (or part of it) is occluded in the cubemap; and (2) in the test scenarios of the cited work, the generations of reflections in scenes with deformable body simulations have not been addressed.

Therefore, this work [1] [3] presents an innovative methodology of rendering at interactive rates, combining reflections in screen space [4] with ray tracing in GPU. In addition, as a proof of concept, it also presents a hybrid rendering engine,

[1]This paper summarizes the results of a PhD dissertation

named VTracer, capable of generating realistic visual effects (reflections and shadows) in static and dynamic 3D scenes.

II. Hypothesis and Research Questions

Considering the current hardware and GPU programming, this work focused on the verification of the following hypothesis:

"The combination of a raster technique and the ray tracing algorithm, in a hybrid way, generates realistic visual effects (shadows and, particularly, reflections) at interactive rates, both in static 3D scenes containing flat and curved objects, and in dynamic 3D scenes, whether these objects are rigid or deformable."

Based on the above hypothesis, six research questions were elaborated and answered (resulting in publications in qualified vehicles), as follows:

1) Is it possible to generate shadows and use a hybrid solution for SSR and ray tracing to generate reflections in 3D scenes at interactive rates? [5], [6], [7] ✓
2) Is it possible to generate realistic reflections of flat and curved objects, whether static or dynamic, through a hybrid rendering solution? [6], [7] ✓
3) What are the existing GPU libraries and acceleration data structures that would support a hybrid solution for reflection generations on dynamic objects in 3D scenes? [5], [8], [7], [9] ✓
4) What are the major acceleration data structures and their update techniques that can be optimized to achieve interactive rates with ray tracing in dynamic 3D scenes? [8], [7], [9] ✓
5) Is it possible to generate realistic reflections at interactive rates, in dynamic 3D scenes containing rigid and deformable objects? [8], [9] ✓
6) Is it possible to evaluate the hybrid solution in different test scenarios, including rigid and deformable body dynamics simulations, focusing on the resulting images's scalability, visual realism, processing time, and frame rate? [9] ✓

III. An Innovative Methodology for Hybrid Rendering to Generate Reflections

Figure 1 shows the architecture and the components from VTracer, being two of them responsible for the special effects: (1) the shadow composer, used for the generation of shadows using shadow maps [10] or ray tracing; and (2) the reflection composer, in charge of combining the reflections from SSR and ray tracer. More specifically, the reflection composer is divided into two steps: (1) one using the SSR [4] algorithm; and (2) another one using a ray tracer.

In the first step, our algorithm gets a reflection map from the G-buffer (*i.e.*, a map indicating which pixels represent objects with reflections) and uses this information to generate the initial reflections applying the SSR. To take advantage of the main benefits that both techniques offer and bypass their limitations, we initially generate both the reflections using the SSR and a mask map of the areas with the scene points in which the SSR failed to calculate the reflections. This is particularly a well-known problem of the SSR, since it only processes what is visible in the image space. The second step of the algorithm focuses on the 3D scene geometry, the mask map containing the pixels that needs ray tracing from SSR and the G-buffer information, wich are passed to the ray tracer to calculate the secondary rays for the reflections (the information referring to the primary rays are already available in the G-Buffer), and then merge the SSR reflection from the previous step with the additional raytraced reflections. The ray tracer verifies each pixel of the mask generated during the first step, searching for pixels that were not correctly calculated. If any pixel satisfying this situation is found, a

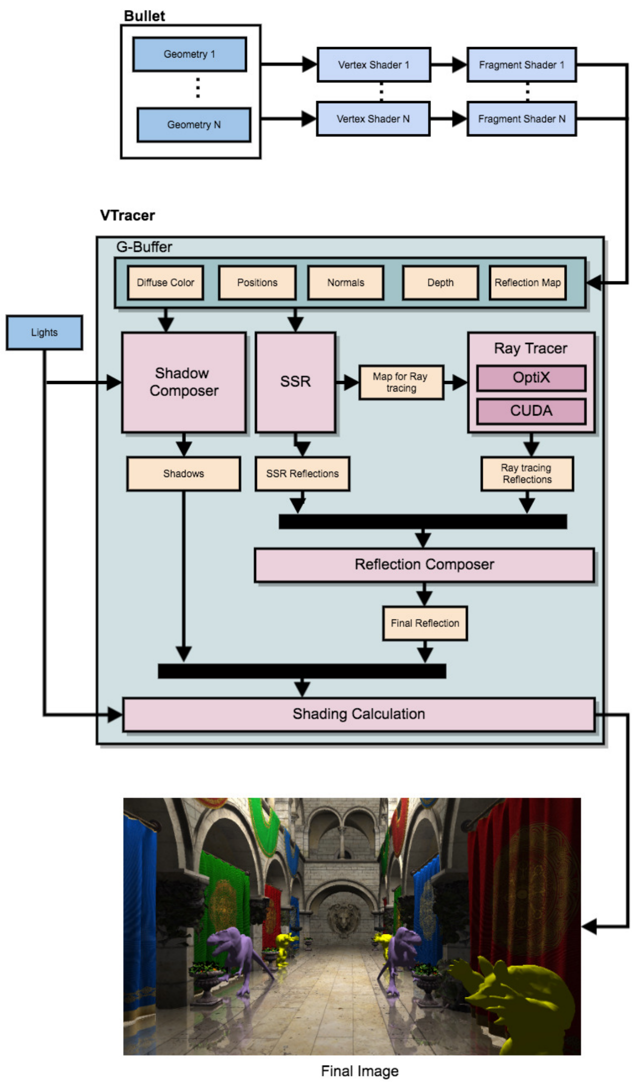

Figure 1: Diagram showing the architecture and the main VTracer's components.

reflected ray from the viewer's eye to that point is calculated and ray traced, resulting in a new complementary reflection image. If necessary, more secondary rays are generated until the reflected object is found. As a result, the reflection of objects that lie outside the screen-space or that are occluded by any other objects can be processed.

It is important to emphasize that, for the ray tracer of the second step, two reflection modules were developed. The first one, with NVIDIA's Optix (used in the first implementation of the algorithm [7]). The second one, in CUDA, based on a GPU data structure inspired by a Linear Bounding Volume Hierarchy (LBVH) [11], to accelerate the generation of reflections in dynamic scenes with rigid and/or deformable objects. We remark that for aiming at processing acceleration, our ray tracer distinguishes between the static objects and dynamic ones, which are divided into two distinct lists, where each list is traversed separately and the nearest intersection point is used, discarding the need of rebuilding the whole data structure of the scene on every frame. Finally, the two reflection images are combined and a Gaussian Blur [12] filter is applied to minimize the artifacts generated by the SSR, especially in regions of transition between the reflections.

IV. Tests, Metrics, and Results

The new methodology presented in this work was applied in varied scenarios. Thorough functional tests, our aim was to identify and correct punctual situations that potencially could generate failures. In addition, we conducted performance tests focusing on realism, frame rate, and level of scalability of the generated images.

To measure the level of realism of the images, we applied two objective metrics (SSIM and RMSE) [13], [14], to evidence the quality of the images. The main diference between these metrics is that the RMSE estimates absolute errors; whereas the SSIM is a perception-based model which considers image degradation as a perceived change in the structural information, while also incorporating important perceptual phenomena, such as both luminance and contrast masking terms. Some test results with different types of reflective objects are shown in Figure 2. Others, performed with dynamic simulations of rigid and deformable bodies, are shown in Figures 3 and 4, respectively. A test including multiple recursive rays in a dynamic simulation of reflective rigid bodies is shown in Figure 5. An illustrative animation of several test results can be visualized at https://goo.gl/NxkEzp

The results show that it is possible to combine reflections in screen space with ray tracing in GPU. Additionally, they show that shadows and hybrid reflections using SSR and ray tracing can be generated in static scenes at high frame rates, usually, over 30 FPS. Regarding to the generated shadows, performance tests were conducted with shadow maps, which achieved much higher frame rates and a visual realism very similar to the results generated by ray tracing.

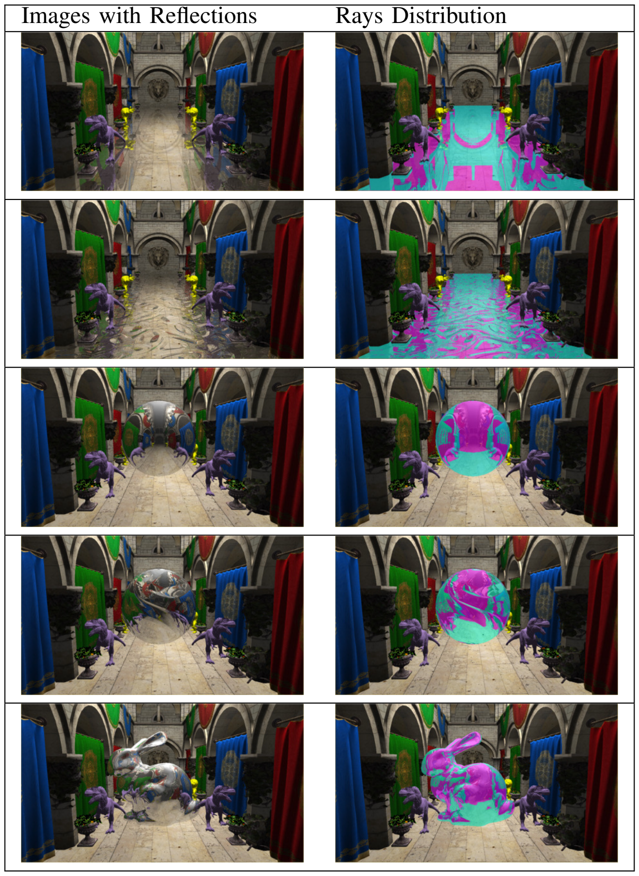

Figure 2: Images generated by the VTracer's hybrid reflection module. Tests with different types of reflective objects, highlighting the ray distribution (colored in pink and blue, respectively, indicating the rendering by ray tracing and SSR).

Several test cases with a realistic hybrid reflection within a static scene with rigid objects which are in constant movement throughout the animation were also successfully produced. In addition, it was possible to conclude that OptiX and its acceleration data structures has not presented satisfactory performance for this type of scenario. This verification motivated the development of a new solution in VTracer, this time fully developed in CUDA, which was successfully used in dynamic scenes testing, as shown in Tables I and II.

Scalability tests of the solution also pointed out that for a scene with many dynamic objects, the realistic rendering in real-time is still very challenging, due to the high computational cost required to update the acceleration data structures of the ray tracing module.

V. Conclusions

The use of ray tracing in graphical applications at interactive rates is becoming more and more a reality mainly due to the advances in algorithms, acceleration data structures, and

Table I: Summarized comparative results of each test run with a dynamic simulation of rigid Stanford Bunnies using CUDA and OptiX

# Stanford Bunnies	**# Triangle Count ($x10^3$)**	**# Dynamic Triangles ($x10^3$)**	**Time mean +/- std dev (ms)**	**Time min, max (ms)**	**Implementation**
5	3.2	2.5	15.2 +/- 0.7	14.0, 17.7	CUDA
10	5.7	5	18.4 +/- 1.1	16.1, 21.5	CUDA
15	8.2	7.5	22.6 +/- 1.2	21.0, 26.5	CUDA
20	10.7	10	28.9 +/- 1.4	25.5, 32.4	CUDA
25	13.2	12.5	31.7 +/- 1.6	29.6, 37.7	CUDA
30	15.7	15	36.5 +/- 1.5	34.4, 43.7	CUDA
5	3.2	2.5	39.1 +/- 2.1	36.2, 46.5	OptiX
10	5.7	5	55.1 +/- 2.6	52.7, 63.5	OptiX
15	8.2	7.5	75.7 +/- 3.7	70.6, 94.0	OptiX
20	10.7	10	98.6 +/- 3.7	88.2, 113.3	OptiX
25	13.2	12.5	111.1 +/- 5.3	104.0, 135.3	OptiX
30	15.7	15	129.3 +/- 6.8	122.1, 163.9	OptiX

Table II: Summarized comparative results of each test run with a dynamic simulation of deformable Utah Teapots using CUDA and OptiX

# Utah Teapots	**# Triangle Count ($x10^3$)**	**# Dynamic Triangles ($x10^3$)**	**Time mean +/- std dev (ms)**	**Time min, max (ms)**	**Implementation**
5	3	2.8	14.7 +/- 0.8	13.6, 13.6	CUDA
10	5.9	5.7	21.9 +/- 1.2	20.5, 26.5	CUDA
15	8.8	8.6	28.4 +/- 1.1	26.5, 31.1	CUDA
20	11.6	11.4	36.4 +/- 1.5	33.8, 41.5	CUDA
25	14.5	14.3	44.3 +/- 2.0	40.6, 51.7	CUDA
30	17.4	17.2	50.0 +/- 1.9	45.4, 55.1	CUDA
5	3	2.8	40.9 +/- 0.7	39.8, 43.5	OptiX
10	5.9	5.7	59.5 +/- 2.7	56.0, 66.3	OptiX
15	8.8	8.6	80.7 +/- 2.9	76.0, 88.2	OptiX
20	11.6	11.4	103.3 +/- 3.9	95.9, 115.0	OptiX
25	14.5	14.3	117.0 +/- 4.4	113.0, 136.6	OptiX
30	17.4	17.2	138.7 +/- 6.5	132.9, 159.1	OptiX

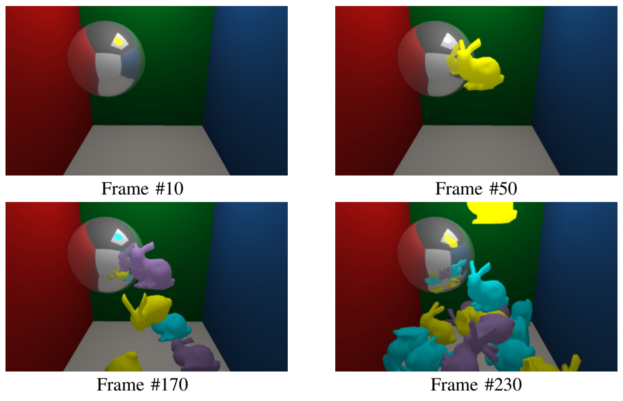

Figure 3: Some animation frames of a dynamic simulation of rigid Stanford Bunnies with a reflective sphere using the ray tracer module in CUDA

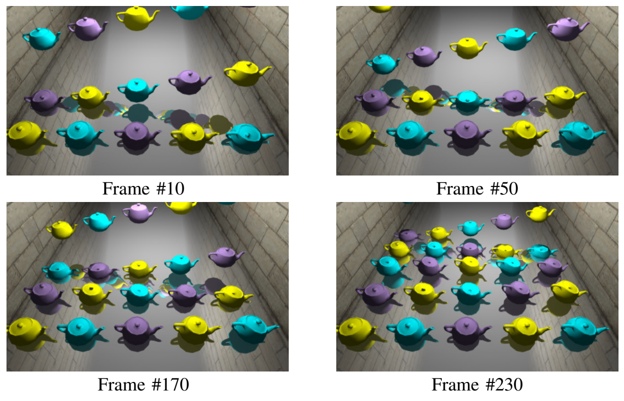

Figure 4: Some animation frames of a dynamic simulation of deformable Utah Teapots with a reflective sphere using the ray tracer module in CUDA.

hardware. In this work, we present an innovative methodology for hybrid rendering of fast and realistic reflections in rigid and deformable body simulations (including multiple recursive rays), with frame rates in general over than 30 FPS. Our solution was implemented for the OptiX library and as a standalone CUDA implementation. When compared to images generated by a complete solution based on ray tracing, the images have a very realistic visual quality with minimal artifacts, which are practically imperceptible. The metric SSIM was helpful to analyze the quality of the dynamic

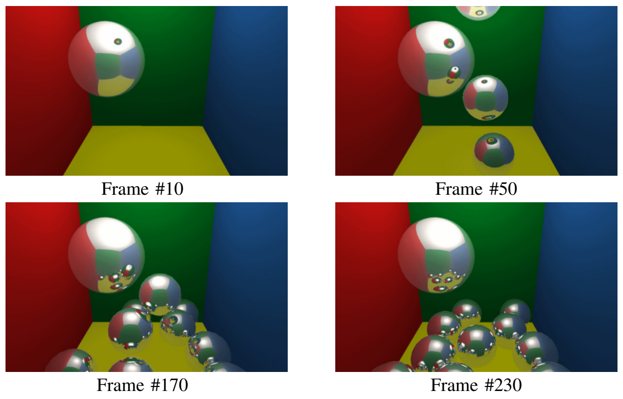

Figure 5: Some animation frames of a test including multiple recursive rays in a dynamic simulation of reflective rigid bodies using the ray tracer module in CUDA.

images along the walkthrough animation in a precise and automated way.

The possibilities of using this solution are many, such as: plugins integrated to game engines, tools for realistic simulation of reflections at interactive rates, a module running together with other library to compose a development kit aimed at realistic rendering, among others. Finally, we expect hybrid solutions, similar to the one presented in this work, to become a trend within the Computer Graphics and that ray traced solutions running at interactive rates, or even in real time, become a reality very soon, either by software, dedicated hardware or both.

VI. Publications

Six publications were directly related to this phd thesis:

1) Fast and Realistic Reflections using Screen Space and GPU Ray Tracing - A Case Study on Rigid and Deformable Body Simulations, ACM Computers in Entertainment, 2018. [9]
2) Real-time Dynamic Reflections for Realistic Rendering of 3D Scenes, The Visual Computer Journal, 2016. [7]
3) Comparison of Acceleration Data Structures for High Quality Fast Reflections of Static and Deformable Models in Walkthrough Animations, SBC Journal on 3D Interactive Systems, 2016. [8]
4) Realistic Rendering in 3D Walkthroughs with High Quality Fast Reflections, in XIV SBGames, 2015. [6]
5) Desenvolvimento de Aplicações Gráficas Interativas com a Unreal Engine 4, RITA, 2015. [15]
6) A Hybrid Rendering Engine Prototype for Generating Real-Time Dynamic Shadows in Computer Games, in XIII SBGames, 2014. [5]

Finally, the following six other publications in qualified Conferences and Journals also contributed to the technical knowledge acquisition and assisted in the study and development of this work: [16], [17], [18], [19], [20], and [21].

Acknowledgment

Daniel Valente de Macedo was supported by CAPES under grant No. 157.257/2012-6 and would like to thank for the financial support.

References

[1] S. Parker, W. Martin, P. Sloan, P. Shirley, B. Smits, and C. Hansen, "Interactive ray tracing," in *Symposium on Interactive 3D Graphics*, ser. I3D '99. ACM, 1999, pp. 119–126.

[2] P. Ganestam and M. Doggett, "Real-time multiply recursive reflections and refractions using hybrid rendering," *The Visual Computer*, vol. 31, no. 10, pp. 1395–1403, 2015.

[3] D. V. Macedo, "Uma nova metodologia para renderização híbrida a taxas interativas combinando reflexões em espaço de tela com ray tracing em gpu," Ph.D. dissertation, Doutorado em Informática Aplicada - Universidade de Fortaleza, 2017.

[4] T. Sousa, N. Kasyan, and N. Schulz, "Secrets of cryengine 3 graphics technology," *Advances in Real-Time Rendering in 3D Graphics and Games Course (SIGGRAPH)*, 2011.

[5] D. V. Macedo and M. A. F. Rodrigues, "A Hybrid Rendering Engine Prototype for Generating Real-Time Dynamic Shadows in Computer Games," in *XIII SBGames, Computação*, 2014, pp. 938–941. [Online]. Available: http://tinyurl.com/jambreg

[6] ——, "Realistic Rendering in 3D Walkthroughs with High Quality Fast Reflections," in *XIV SBGames, Computação*, 2015, pp. 9–15. [Online]. Available: http://tinyurl.com/zeqsl93

[7] ——, "Real-time Dynamic Reflections for Realistic Rendering of 3D Scenes," *The Visual Computer Journal*, vol. 34, no. 3, p. 337–346, 2016. [Online]. Available: https://tinyurl.com/y7nh3hwk

[8] ——, "Comparison of Acceleration Data Structures for High Quality Fast Reflections of Static and Deformable Models in Walkthrough Animations," *SBC Journal on 3D Interactive Systems,*, vol. 7, no. 1, pp. 28–37, 2016. [Online]. Available: http://tinyurl.com/h6xhmur

[9] D. V. Macedo, Y. R. Serpa, and M. A. F. Rodrigues, "Fast and Realistic Reflections using Screen Space and GPU Ray Tracing - A Case Study on Rigid and Deformable Body Simulations," *ACM Computers in Entertainment*, vol. 16, no. 4, 2018.

[10] R. Fernando and M. Kilgard, *The Cg Tutorial - The Definitive Guide to Programmable Real-time Graphics*. Addison-Wesley, 2003.

[11] C. Lauterbach, M. Garland, S. Sengupta, D. Luebke, and D. Manocha, "Fast BVH construction on GPUs," *Computer Graphics Forum*, vol. 28, no. 2, pp. 375–384, 2009.

[12] L. Shapiro and G. Stockman, *Computer Vision*. Prentice Hall, 2001.

[13] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE TIP*, vol. 13, no. 4, pp. 600–612, 2004.

[14] R. Hyndman and A. Koehler, “Another look at measures of forecast accuracy,” *International Journal of Forecasting*, pp. 679–688, 2006.

[15] D. V. Macedo, Y. Serpa, and M. A. F. Rodrigues, “Desenvolvimento de Aplicações Gráficas Interativas com a *Unreal Engine 4*,” *RITA*, vol. 22, no. 2, pp. 181–202, 2015. [Online]. Available: http://tinyurl.com/gs4zbph

[16] D. V. Macedo and M. A. F. Rodrigues, “Experiências com desenvolvimento ágil de um jogo casual para plataformas móveis usando o motor gráfico *Unity*,” in *XI SBGames, Computação*, 2012, pp. 148–156. [Online]. Available: http://tinyurl.com/zt4exbb

[17] D. V. Macedo, H. Pontes, and M. A. F. Rodrigues, “Criação de arte usando um dispositivo háptico para pintura digital interativa,” in *XV Symposium on Virtual and Augmented Reality*, 2013, pp. 288–291. [Online]. Available: http://tinyurl.com/jhnlxv8

[18] M. A. F. Rodrigues, D. V. Macedo, Y. Serpa, and et al., “Combatendo a halitose: Um *Serious Game* multiplataforma em saúde bucal,” in *XIII SBGames, Arte e Design*, 2014, pp. 210–219. [Online]. Available: http://tinyurl.com/h2scjnz

[19] ——, “Um *Serious Game* em saúde bucal contra o tártaro,” in *XIV SBGames, Arte e Design*, 2015, pp. 699–702. [Online]. Available: http://tinyurl.com/j8ao2k4

[20] ——, “Beyond fun: An interactive and educational 3D traffic rules game controlled by non-traditional devices,” in 30^{th} *ACM SAC*, 2015, pp. 239–246. [Online]. Available: http://tinyurl.com/zr3ep6q

[21] M. A. F. Rodrigues, D. V. Macedo, H. P. Pontes, Y. Serpa, and Y. R. Serpa, “A serious game to improve posture and spinal health while having fun,” in 4^{th} *International Conference on Serious Games and Applications for Health (SeGAH 2016)*, 2016, pp. 1–8. [Online]. Available: http://tinyurl.com/zxsraw4