Laje de Santos: a dive site recreated in virtual reality

Bruno Barberi Gnecco
brunobg@corollarium.com
Corollarium Technologies
São Paulo, Brazil

Marcelo de Paiva Guimarães
Paulista School of Politics, Economics and Business (EPPEN/UNIFESP)
Computer Science Master Program (Unifaccamp)
Osasco, SP, Brazil

Figure 1: Screenshot of Laje de Santos, 3D rendered in real time by our application.

ABSTRACT

This project used 2D and 3D graphics to raise awareness and information about the Laje de Santos Marine State Park (*Parque Estadual Marinho da Laje de Santos*), the first marine park of the State of São Paulo in Brazil. The project's main goal was to allow people from all over Brazil and the world to virtually visit the park, since visiting the park requires two hours of navigation on a speed boat and a SCUBA diving certification to see the underwater life. Through our project people can make simulated virtual reality SCUBA dives, learn about the fauna that inhabits the park and the marine environment. We detail in this paper the process to reconstruct the park in 3D and create the VR experience. This project is available on a website with text, images, videos and 3D content that works on any browser with no need of a separate app download, and has its full source code published.

CCS CONCEPTS

• Software and its engineering → Virtual worlds software; **• Human-centered computing → Virtual reality**.

KEYWORDS

virtual reality, SCUBA diving, photogrammetry, webgl, web

ACM Reference Format:
Bruno Barberi Gnecco and Marcelo de Paiva Guimarães. 2022. Laje de Santos: a dive site recreated in virtual reality. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (SVR '22).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

SVR '22, October 03–05, 2022, Natal, Brazil

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The Laje de Santos Marine State Park (*Parque Estadual Marinho da Laje de Santos*, or PEMLS) is the first Marine Park in the Conservation Units of the State of São Paulo. Located about 40km away from the coast, with no other rock formations or islands nearby, it has a great concentration of passing fish and reefs, being one of the main diving and underwater photography sites in Brazil. The Park also has an artificial ship wreck.

The Park can be visited by boat, but docking on the island is forbidden. The navigation is long, about 1h30 to 2h in speedboats or 6h to 8h in slow boats, if the weather allows, which makes the visit difficult, long and expensive. It is basically restricted to accredited autonomous SCUBA divers. As a result, few people have access to a place of enormous beauty and diversity of fauna.

Hence, though the PEMLS is a public area and an important milestone in recognizing that the marine environment must also have environmental protection parks, it is practically unknown outside the diving and nautical communities. The awareness and care of the ecological environment has become increasingly important, but these environments are often physically and culturally distant from population centers. When they are located in the ocean they are even more distant, since most people never have the chance to see the underwater environment due to the extra costs and the requirement of having a diving certificate.

Spreading information about the park, the marine environment and the marine fauna makes the population learn about its existence and raises awareness of its importance. This led to the conception of the project presented in this paper: building a virtual reality visit to PEMLS, complemented by a rich website with text, images, videos and 3D models.

We chose virtual reality because it is a ludic and interesting format, and allows us to experience what scuba diving is like. It is a wonderful technology to bring people to environments that are far way and difficult to reach. We wanted to spread the knowledge of how important it is to have marine parks for the preservation of the sea, species reproduction and migration. There's no greater way to understand this than seeing the richness of those places, and an interactive experience mimicking the real one is as impactful as it gets.

This paper details our implementation, using WebGL2 for the 3D and VR content, and the challenges we faced to make the application run in real time even in mobile devices. We go over the whole process, from underwater image acquisition to 3D reconstruction, the 3D code architecture and challenges for real time realistic rendering and the final application.

2 SIMILAR PROJECTS

There have been several attempts to recreate underwater areas in 3D. They are usually focused on wrecks, which are visually interesting and have historical value. Significant projects include the Thistlegorm Project[5], a project that recreated in 3D through photogrammetry a Second World War ship wreck located in the Red Sea. The British ship was carrying ammunition, cars, motorbikes, guns and even a locomotive, when it was bombed by a German plane and sunk immediately. The ship was later found by Jacques Cousteau and is one of the most famous dive sites in the world. The Black Sea MAP (Maritime Archaeological Project) [10] is an international project that researchs and maps in video and 3D wrecks in the Black Sea.

Several other attempts to recreate wrecks in 3D from photogrammetry were made, but we are not aware of any similar projects for the 3D reconstruction of an entire dive site in Brazil.

3 THE LAJE DE SANTOS MARINE STATE PARK

The Laje de Santos Marine State Park is about 22 nautical miles (40km) away from the cities of Santos or São Vicente[3]. It was created on September 27, 1993, through State Decree nº 37.537, with the objective of protecting marine biodiversity and conserving biological diversity on the coast of the State of São Paulo.

It includes the rocky islet of Laje de Santos itself, as well as the parcels of Bandolim, das Âncoras, Brilhante, do Sul and Novo and the rocks known as Calhaus, totaling 5,000 hectares of preserved area in a rectangle shape (10,000 by 5,000m). The Laje de Santos islet has a shape reminiscent of a whale, 550m long, 33m high and 185m wide. A Brazilian Navy signal lighthouse is located at its highest point. It has practically no vegetation and is home to a large number of seabirds, which use the site as a breeding and resting area; examples of these birds are: brown boobies, terns and the gull.

Figure 2: Split level photo of Laje de Santos during one of our dives. The lighthouse can be seen on the island, as well as some birds and one of our divers underwater.

Several species that occur in the PEMLS are on the list of Brazilian fauna threatened with extinction. The Park is part of the route of several migratory species such as whales, dolphins and seabirds. It is an important feeding area for several species, many of which are protected by international conventions, such as Bryde's whales, manta rays, sea turtles and others.

In addition to the other points mentioned, the PEMLS is known to be on the migratory route of manta rays. The manta ray is one of the largest fish in the world, reaching eight meters in wingspan and weighing more than two tons. The *Mantas do Brasil* project seeks to expand knowledge in order to preserve the largest stingray species in the world, especially the species proven to be incident in Brazil. In 2011 both of the world's described manta species were re-categorized, leaving the former "NEAR THREATENED" stage to the more worrisome "VULNERABLE TO EXTINCTION" stage on the IUCN red list. Brazil is suspected to have the smallest manta ray population in the world, with a population estimate of only a few hundred individuals.

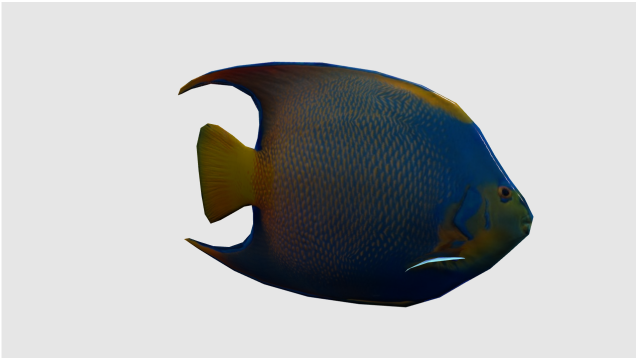

Figure 3: Our 3D model for the angel fish (*Holocanthus ciliaris*).

As the PEMLS is one of the main diving and underwater photography spots in the country, the main regulated activity within the Park area is the practice of free and autonomous diving. Therefore, the current management plan for public use is aimed at boat navigation for tourism companies and accredited diving operators. On July 1, 2003, Instituto Laje Viva was created, a non-governmental organization (NGO) with the aim of helping to protect the fauna and environment of the Laje de Santos Marine State Park.

PEMLS is the least visited park in the State of São Paulo. The average numbers between 2004 and 2013, the latest available, are only 3,433 visitors per year, with a peak of 4,518 visitors in 2007. To put that into perspective, it's less than 10 times the visitation of the Museu da Pesca (Fishing Museum) in Santos.

PEMLS also houses the first artificial wreck in Brazil, a boat popularly known as Moreia. Moreia was wrecked on purpose in 1992, and for a long time almost nobody knew who was responsible for it. The story only became public when the documentary Laje dos Sonhos [11] was released, in which the responsible person describes how it all happened. He found a boat hull and towed it to the island. He thought the plunge would be almost immediate, but it took hours for the boat to sink. Slowly dismantling due to the action of water and time, the Moreia remains at the bottom providing a beautiful visit for divers. One of the goals of this project was to capture the Moreia in 3D for posterity. Reports are that storm surges have already moved the shipwreck several times – it seems that its initial position was not even its current one, but lying down. It's bound to collapse in the next few years, as significant damage to the deck has already happened and penetration is currently forbidden.

4 IMPLEMENTATION

This project required a lot of technology to get off the ground – or under the water, in this case. Its flow included from image capture to 3D reconstruction, from innovative web design to a 3D web implementation that had to be fast even on mobile devices.

The project started in 2020, with the development of the 3D software base, the website and its content and the planning of the dives. The start of the dives was scheduled for the end of March 2020, when lockdown took place due to the COVID-19 pandemic. The Park was closed and any diving activity, being done on boats that do not allow for social distancing, was severely affected.

We made a deliberate choice to implement the entire code in a web-friendly format. We wanted zero-friction to access and use the application, so anyone with a modern web browser would be able to experience the application. Considering also international users, we considered internationalization from the start.

Most modern web browsers implement WebGL2, an immediate 3D rendering API designed for the web [6]. As WebGL2 is a low level API, we studied the two most popular web 3D engines, ThreeJS [13] and BabylonJS [8]. While ThreeJS was more widely used, we ran into issues during the caustic implementation, and moved our code base to BabylonJS. This will ensure future compatibility with the new WebGPU standard [9].

4.1 Reconstruction

We made extensive use of photos in this project to capture data. All animals were hand modeled in Blender [2] from pictures we took. This ensured a beautiful mesh in a simple low-poly version, as well as proper animations made with bones.

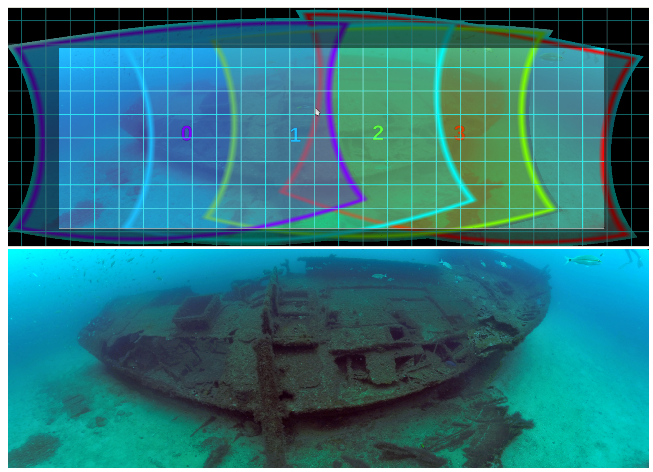

Figure 4: Four images used to compose a full size picture of Moreia. They were distorted to compensate for the camera lens and the images were composed together.

Other things were more complicated. The island above the water was easier to reconstruct from existing satellite height data, but underwater we only had rough depth lines from navigation maps. We used our own knowledge to make a rough draft of it. It was unfeasible, given the time and money constraints of this project, to photograph the entire island underwater to reconstruct it with photogrammetry. That would have mean tens of thousands of photos, if not hundreds of thousands. We used hand modelled rocks textured from actual underwater photos.

Due to sediments, plankton, algae and other things, we usually can only see a few meters away underwater. At Laje the Santos the usual visibility is between 5m and 12m, and only in rare occasions you can see up to 20m. Even on these perfect days, 20m means that "at 20m you cannot see anything anymore", with visibility decaying exponentially before that point. Moreia, the wreck, is about 22m long, so it's almost to impossible to take a picture covering all of it. If you are close enough to get a clear picture, you are too close

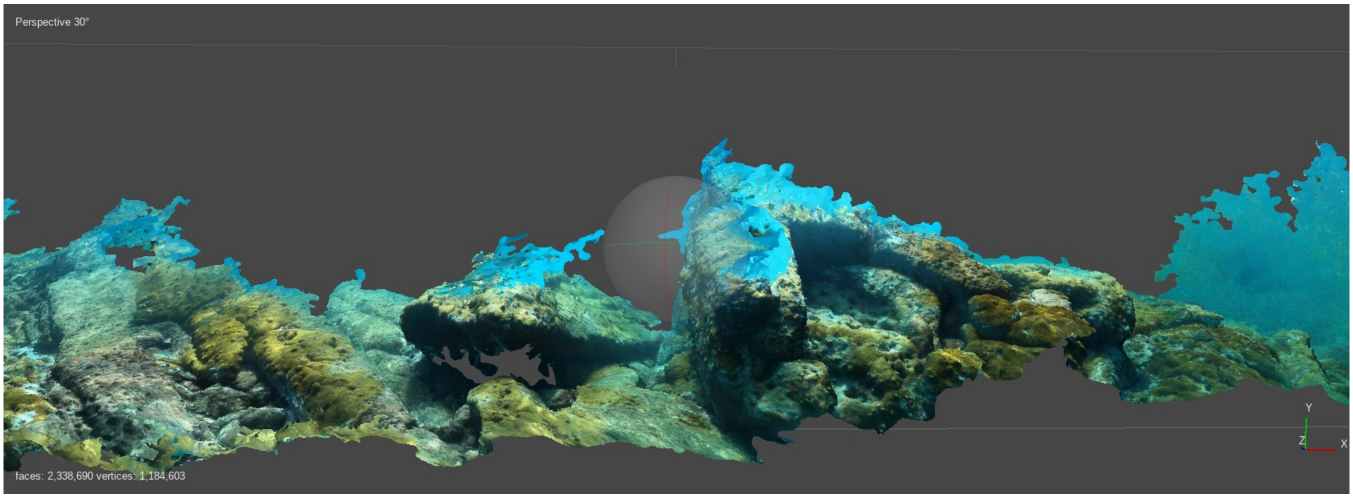

Figure 5: Partial photogrammetric reconstruction of a single row of photo captures to recreate the underwater landscape in 3D.

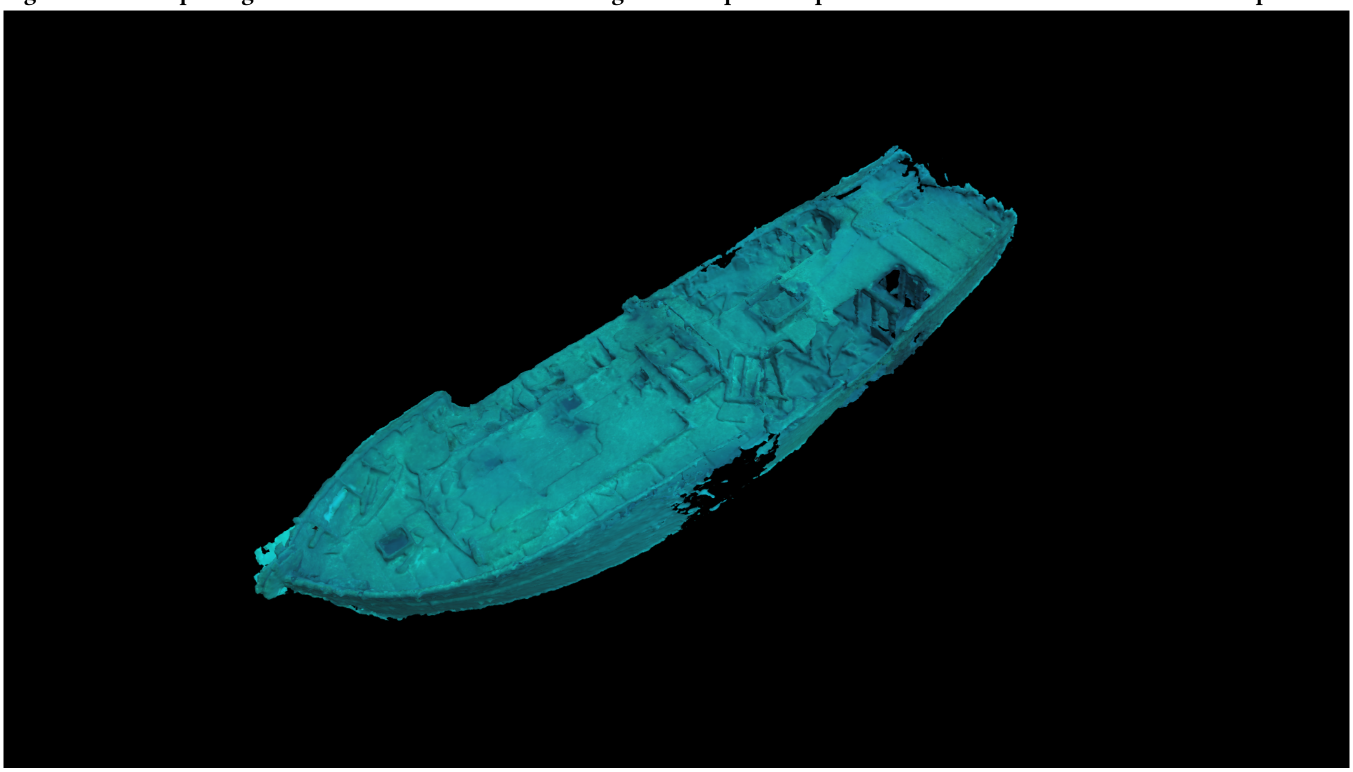

Figure 6: Our 3D reconstruction of the Moreia ship wreck.

to get the entire wreck into a single picture, even with a wide lens. Figure 4 shows a 2D composition of four images that simulate what would be visible it if were possible to photograph the wreck from a single spot and full visibility.

For 3D reconstruction, though, we considered photogrammetry. Photogrammetry consists in algorithms to reconstruct a 3D model based solely on photos. While algorithms take care of the heavy work, there's still some art in taking the pictures properly and making some decisions for the software. We tested several software packages for photogrammetry, but settled on Agisoft Metashape.

Photogrammetry is a resource intensive algorithm, however. A large amount of photos needs to be taken, covering the entire area with significant overlap between the pictures so their relative positioning can be recreated. Figure 5 shows a 3D reconstruction of the underwater landscape, which used 129 photos. It was unfeasible to capture the whole underwater area, or even a significant part of it. We ended up recreating it from approximate water depth maps and manually modeled rocks.

The Moreia wreck needed careful 3D reconstruction, though, with realistic data. It would be unfeasible to do it properly with manual 3D modeling, and in its case we used photogrammetry. We took pictures of the Moreia on more than one dive, and the final reconstruction was based on 734 photos taken by us underwater. It ran in two servers, one with 16 CPUs and a GPU for the photo alignment and dense cloud building, and another with 150GB RAM for the memory intensive mesh reconstruction, resulting in a mesh with 37 million polygons that was later decimated to 500k, which renders in real time even in mobile devices, as seen on figure 6.

4.2 Underwater rendering

Things look different underwater. First, water absorbs light, and does it according to frequency. Red light is almost completely absorbed in the first few meters. Around 20 meters deep, which is the average depth around Laje, you're left essentially with green and blue light. This is why underwater pictures are very blueish. Underwater pictures are often taken with a light strobe for close-ups, which makes them "true color", but even in that case far away objects are not lit due to the same absorption rates. Underwater photos are also heavily color corrected on software to make them look "more colorful", usually through white balancing, color curves, contrast and saturation.

It's interesting, however, that our brains compensate for that color loss and when we're actually diving we barely notice the lack of red. Underwater things look "blueish but realistic", much different from the raw images we see afterwards on camera.

Besides this blueish coloring, water has caustics, the light effect seen at the bottom of pools, where light seems to dance in chaotic patterns. This happens due to refraction of the light on the air-water interface. It's a pretty effect and one that our eyes immediately connect with underwater. One can easily see it on shallow dives and sunny days. It's close to impossible to accurately simulate caustics in real time; calculating the actual light transmission is very expensive computationally, and requires a global illumination algorithm, often implemented with ray tracing. The usual way to do handle it is to fake it through an animated noisy texture projected on the meshes. Rendering [4, 7, 12] it on top of actual textures, however, presents a challenge: you need to multiply the caustic texture by the color texture, taking also in consideration the angle between the light direction and the mesh normal. Another light effect when diving are "god rays", or volumetric light that comes from scattering of particles, just like a dusty room makes light look like a 3D object.

One way to implement all of this is through multipass rendering. This was our initial implementation, which included some contributions to the BabylonJS project. The algorithm is to render the scene twice, once with regular textures, and another swapping the textures for an animated caustic shader. We implemented a noise algorithm, mapping the UV coordinates to the XZ plane (normal to the up vector), and multiplying the caustic color by the dot product of the mesh normal and the up vector, clamping to only positive values (since caustics are caused essentially by direct light and cannot be seen on the bottom of meshes). Caustics have also to be restricted

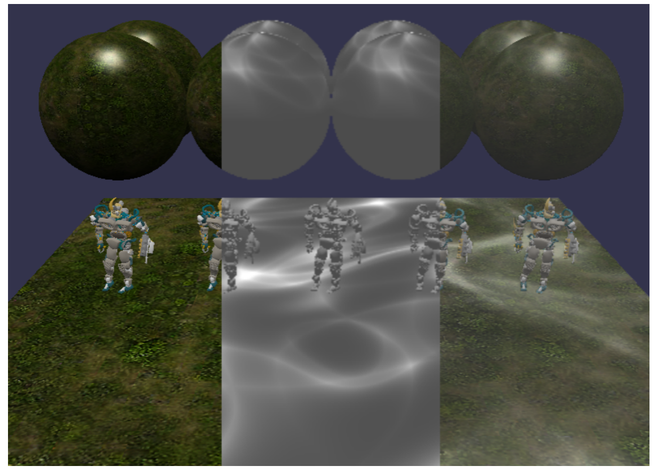

Figure 7: Our multipass approach for caustics. On the left the real color rendering. On the middle the caustic render, which is done at a slightly lower resolution to improve performance. On the right the images are composed for the final effect.

only to underwater areas. Once these two passes are rendered to two target textures, we multiply them on post processing to apply the caustic to the color texture. Figure 7 show the algorithm at work.

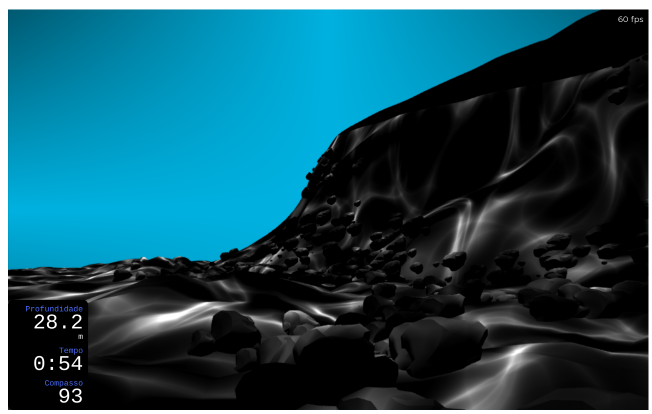

Figure 8: Pure caustic render of the scene. Notice that areas beyond the Z-far plane escape (rendered in blue) and that areas above water have no caustics (pure black).

The problem with multipass rendering, however, is that there's a noticeable performance loss, particularly on WebGL. Modern native engines often do several passes per frame, but even two passes on WebGL tend to affect the frame rate. To handle the performance loss we implemented a different approach, Material Plugins. This approach means patching the fragment and vertex shader code when they are being compiled, inserting code that computes the necessary data and modifies the resulting pixel color from the fragment shader, applying the caustic texture and underwater absorption effect. Figure 8 shows only the caustic effect running on our scene terrain, with all other effects and texturing turned off.

We patched BabylonJS to allow for Material Plugins together with the project developers. This was a considerable change that was incorporated into BabylonJS and increased its flexibility, being listed as one of the major changes in its v5.0 release. This resulted in dropping the second pass, moving back to a single render, with negligible performance loss compared to the non-caustic rendering, since we only added a few instructions to the shader code, and even better effects, since we don't render the caustics at a lower resolution.

4.3 Vertex Animation Textures

The idea of having huge schools of fish was an absolute necessity to this project. It's common to see schools with dozens of fish, particularly salemas and sargentinhos. This meant we'd need not only to render hundreds of fish, but animate them. Both are demanding tasks in WebGL, since the naïve way to render them (unique meshes, individual animations) is CPU heavy.

BabylonJS has two ways to handle many copies of the same mesh, with thin instances and particles. This was used to render thousands of rocks of our island model, but the fish animation was still slow, since BabylonJS applies bone animations to the mesh on the CPU. However, we can pre-bake the animation with Vertex Animation Textures (VAT). The idea is to pre-generate the matrix transformations for all vertices and frames of the animations, saving them to a texture. Then not only you don't have to compute the animation in real time anymore, but you also can use a vertex shader to apply it on the GPU. Each instance has unique variables, such as the animation speed, initial offset and current time, so their animation is not synchronized. Figure 9 shows how this enables hundreds of instances at the highest possible frame rate.

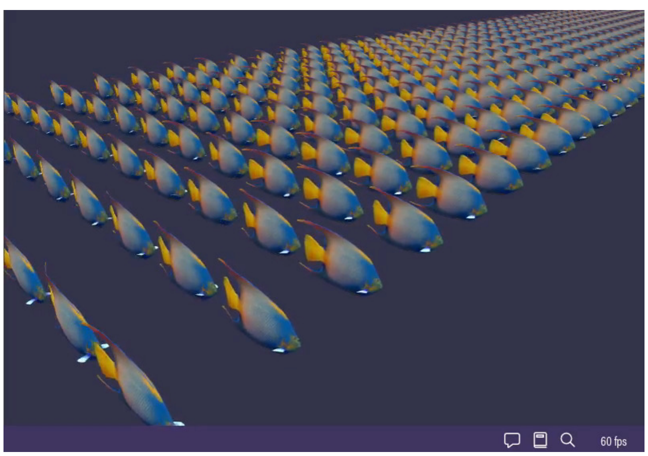

Figure 9: Vertex Animation Textures showing realtime rendering of hundreds of animated models in WebGL.

There was no support for VAT in BabylonJS, other than a proof of concept demonstration. We implemented it (again, with help from the BabylonJS community) and it also became part of the BabylonJS package.

4.4 Ocean rendering

Large scale water rendering presents challenges. Rendering a large plane mesh is not scalable. One approach is to adapt the mesh to have larger density closer to the screen, but that requires code to handle the level of detail dynamically, has edge cases that have to be treated on the boundaries between density changes, as well as generating artifacts with the water simulation. All of this is complicated on WebGL since the current JS model implemented on browsers has severe limitations for multi-threading.

Another popular approach to this problem is to do the rendering of water in post-processing. This is scalable since we only have to check the rendered pixels, not the full size of the mesh, and have a quick early release on the algorithm for pixels that definitely don't hit the ocean. The implementation consists in a ray marching algorithm, which is implemented in the GPU shader, making it fast and avoiding the usual CPU bottleneck on JS applications [1]. For each pixel a ray is marched, checking whether it hits the ocean.

The idea is that the water is close to a plane, but with perturbations in height generated by an animated heightmap. The waves are a composition of sin/cos functions with non-integral exponentials (see figure 10). This looks choppy and wavy enough, and by composing different frequencies and amplitudes the result is very much like a real ocean. This allows one to compute the ocean height at most a fixed number of times, and the height is controlled by a noisy water-like simulation, which is also much faster than solving a physically correct model such as Navier-Stokes.

The ocean shader in this project uses a step size proportional to the ratio between a near and far pair of points for the marching phase. These are initially the camera and "infinity", and the extremes are updated iteractively, with this adaptive step size, for a fixed number of steps.

Our shader code extends [1] in many ways, including a proper way to handle intersections if the camera is below the $y = 0$ plane. We wrote fragment shader color code to make it render a more accurate representation of what you see underwater when you look up. This includes the so called Snell's window, which is the area where you can see the surface because the light is partially refracted. Outside this area there's total internal reflection. The practical result is a "circle of light" permanently hovering above the person. We initially considered implementing this as a fixed plane above the camera, but the ray marching version gives a better underwater view. Snell's window has strong refraction and works pretty much like a wide angle lens. Refraction based on the normal to the surface, mapped to a spherical textured sky resulted in a very realistic effect, as shown on figure 11. We used the same shader for rendering the sky above water as well, when the ray escapes to infinity with a positive y.

4.5 Diving simulator

The core piece of the project is a SCUBA diving simulator that allows people to experience a simplified version of what diving is like. SCUBA diving requires learning some concepts in order to be done safely and how to use the autonomous diving gear. We decided to incorporate only the basic behavior, not forcing the user through a learning tutorial for a experience that will last only

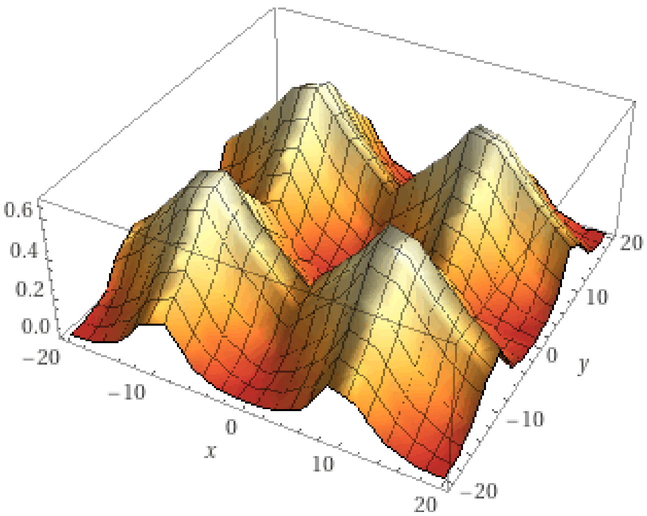

Figure 10: A plot of the wave function used for the ocean. Summing multiple instances of this wave function with different frequencies and amplitudes results in realistic animated water, but fast enough to be rendered in real time during post processing.

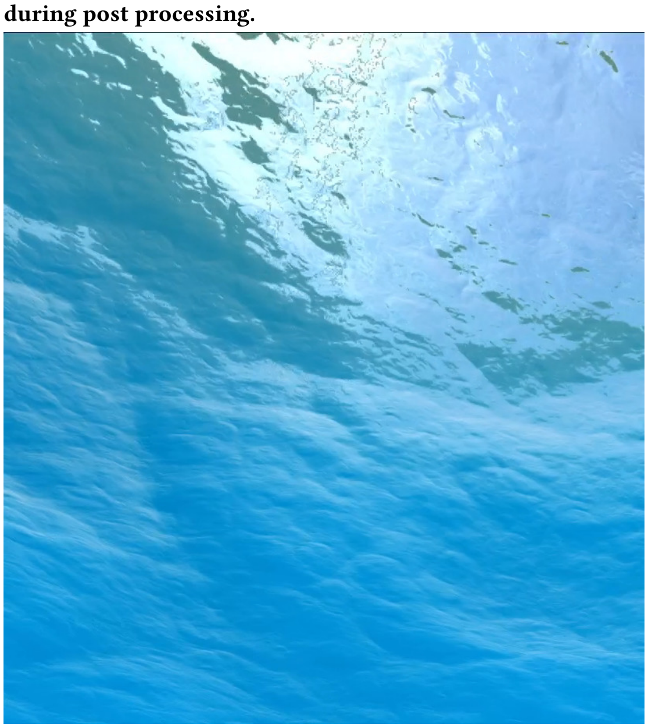

Figure 11: Real time 3D rendering of the ocean as seen from underwater in our application, showing the Snell window effect on the top right, where some of the sky is visible.

minutes. However we did implement a dive computer on the GUI, which shows depth, dive time, compass and ascent rate.

We added audio to complement the experience. Diving has a particular sound of breathing underwater that is part of the experience. The Moreia wreck was positioned in its corresponding geographic location, and we added some animated fauna: turtles, manta rays, fish.

4.6 Complementary information

The VR simulator is part of a larger website about the PEMLS, which includes information about its fauna and flora, the dive sites, the park history and our project. 3D content is interspersed on the website together with images and videos, such as models of the fishes. Several areas of the webpage are actually rendered in 3D, such as the dive effect of the initial page. WebGL has brought 3D content as another media that is easily integrated, giving developers more options than just only a stand-alone 3D experience.

5 CONCLUSION

The main objective of this project was to bring a marine park to the public through a virtual platform on the web, allowing it to be seen on any device with an internet browser, without having to install any software or application. Thus, we see it as a service to the society, providing information, videos, images and the 3D diving simulator to be able to have an experience that very few people currently can do.

We are also in the process of installing a virtual reality environment at the Santos Fishing Museum. The environment includes a submarine model, which is being refurbished for the project.

As future work we plan to extend the data acquired and attempt a larger size photogrammetric reconstruction, as well as checking new NeRF (Neural Radiance Fields) representation for view synthesis. We're also waiting for the larger adoption of WebGPU and WebXR technologies, which would allow us to achieve better performance and improve our VR integration.

ACKNOWLEDGMENTS

This project was partially supported by ProAC do Estado de São Paulo in 2020.

REFERENCES

[1] Alexander Alekseev. 2014. Seascape. https://www.shadertoy.com/view/Ms2SD1
[2] Oscar Baechler and Xury Greer. 2020. *Blender 3D By Example: A project-based guide to learning the latest Blender 3D, EEVEE rendering engine, and Grease Pencil, 2nd Edition* (illustrated edition ed.). Packt Publishing, Birmingham, Mumbai.
[3] Fundação Florestal de São Paulo. 2017. *PLANO EMERGENCIAL DE USO PúBLICO. Anexo I da Portaria Normativa FF/DE nº 257 de 12/06/2017.* Technical Report. Fundação Florestal de São Paulo. http://arquivos.ambiente.sp.gov.br/fundacaoflorestal/2017/06/2017-Port-FF-257_Anexo-I-_PEUP-PEMLS-vfinal.pdf
[4] Eric Haines, Naty Hoffman, and Tomas Akenine-Mo]ller. 2018. *Real-Time Rendering, Fourth Edition* (4th ed. edição ed.). A K PETERS, Boca Raton.
[5] Jon Henderson. 2017. The Thistlegorm Project. https://thethistlegormproject.com/ Accessed: 2022-08-01.
[6] Dean Jackson and Jeff Gilbert. 2022. *WebGL2 Specification.* Technical Report. https://registry.khronos.org/webgl/specs/latest/2.0/
[7] Eric Lengyel. 2019. *Foundations of Game Engine Development, Volume 2: Rendering.* Terathon Software LLC, Lincoln, California.
[8] Microsoft. 2022. BabylonJS. https://babylonjs.com/
[9] Kai Ninomiya, Brandon Jones, and Dzmitry Malyshau. 2022. *WebGPU.* W3C Working Draft. W3C. https://www.w3.org/TR/2022/WD-webgpu-20220728/.
[10] University of Southampton. 2022. Black Sea MAP (Maritime Archaeological Project). https://cma.soton.ac.uk/research/black-sea-map/ Accessed: 2022-08-01.
[11] Rose Bezerra; Raquel Pellegrini. 2012. Laje dos Sonhos. https://www.youtube.com/watch?v=emBUPoyJ8Pk

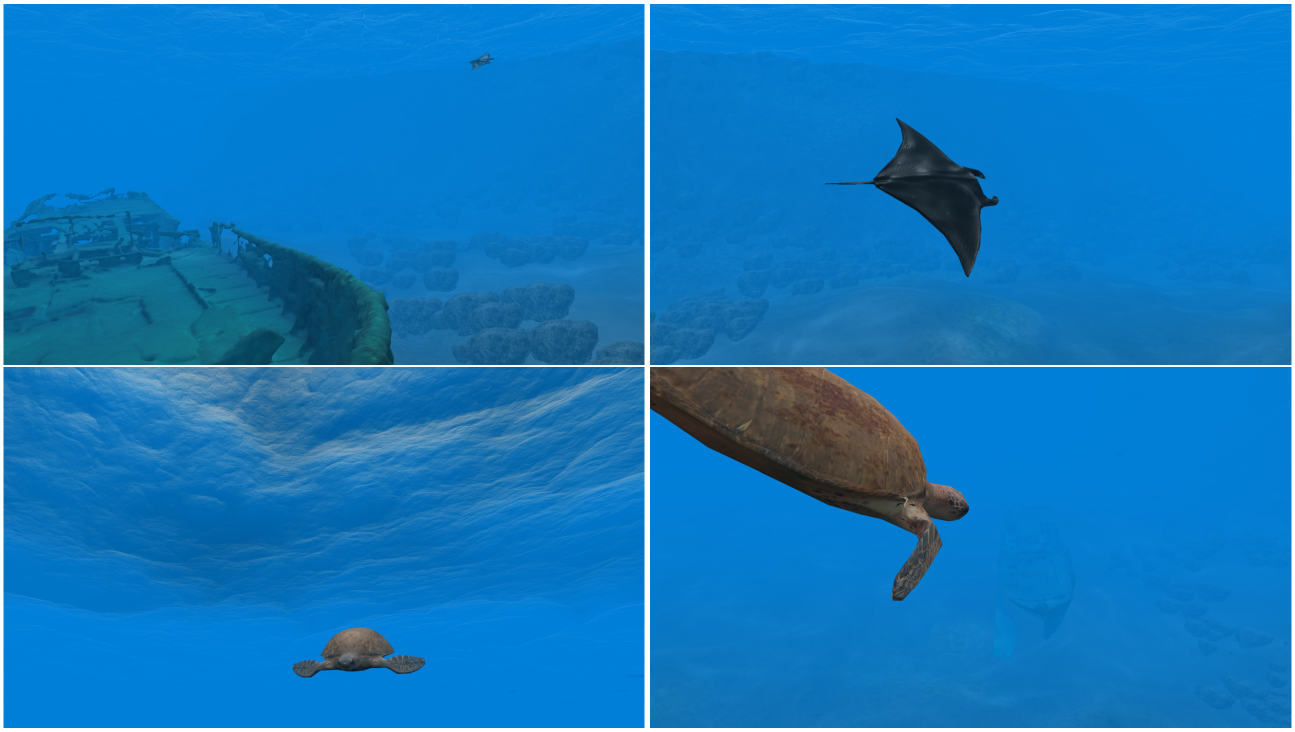

Figure 12: Screenshots of the dive simulator.

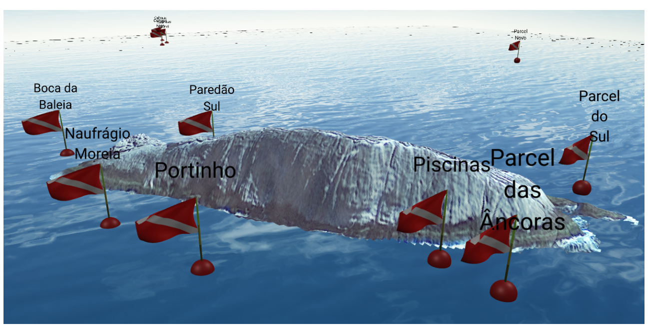

Figure 13: A 3D interactive map of the dive sites in the PEMLS, focused on Laje de Santos. On the back one can see Calhaus and Parcel Novo.

[12] Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically Based Rendering: From Theory to Implementation* (3rd ed. edição ed.). Morgan Kaufmann Publishers, Cambridge, MA.

[13] three.js. 2022. ThreeJS. https://threejs.org/