Tekhne e Logos, Botucatu, SP, v.10, n.2, set., 2019.
ISSN 2176 – 4808

DESENVOLVIMENTO DE APLICAÇÃO EM REALIDADE VIRTUAL

DEVELOPING APPLICATION ON VIRTUAL REALITY

Jorge Henrique Faine Monteiro[1] Gustavo Kimura Montanha[2]

RESUMO

A realidade virtual e aumentada é uma tecnologia que, a cada dia, ganha mais espaço nos mais diversos segmentos dos negócios. Frente ao competitivo cenário do mercado, empresas têm adotado cada vez mais estratégias envolvendo essa tecnologia. Este artigo demonstra o desenvolvimento de uma aplicação em realidade virtual que simule um ambiente subaquático para aplicação em ambientes atrativos como shoppings. Para seu desenvolvimento, foi utilizado o *software* Unity, responsável pela criação e ambientação em 3D e os Oculus Rift DK2 para visualizar sua execução. O aplicativo obteve o resultado desejado, onde os usuários simularam o ambiente subaquático sem apresentar sintomas de enjoos. Com a utilização das técnicas de otimização, foi possível obter cenas mais complexas sem prejudicar o processamento do equipamento.

Palavras-chave: Oculos Rift. Realidade Virtual. RV. Unity3D.

ABSTRACT

Virtual and augmented reality is a technology that is daily growing and is expanding to the most diverse segments of the business. Considering the competitive market scenario, companies have increasingly adopted strategies involving this technology. This paper shows the development on virtual reality application that simulates an underwater environment for application in attractive environments such as malls. It was used Unity software, responsible for 3D creation and ambiance and the Oculus Rift DK2 to visualize its execution. Obtained results were positive, showing that users simulated the underwater environment without experiencing motions sickness. Using optimization techniques, it was possible to obtain more complex scenes without harming equipment processing.

Keywords: Oculus Rift. Virtual Reality. VR. Unity3D.

[1] Aluno de graduação do ADS – Faculdade de Tecnologia de Botucatu
[2] Docente da Faculdade de Tecnologia de Botucatu – SP. Av. José Ítalo Bacchi, S/N. Jardim Aeroporto, Botucatu – SP. Email. gmontanha@fatecbt.edu.br

Recebido em novembro de 2018 e aceito em maio de 2019.

Tekhne e Logos, Botucatu, SP, v.10, n.2, set., 2019.
ISSN 2176 – 4808

1 INTRODUÇÃO

A Realidade Virtual é uma tecnologia que permite a imersão em um mundo totalmente virtualizado, possibilitando uma melhor compreensão e interação desse mundo gerado pela tecnologia. Segundo KIRNER e TORI (2004), o termo Realidade Virtual (RV) foi criado no final da década de 1980 por Jaron Lanier que possibilitou juntar dois conceitos antagônicos em um novo, capaz de definir melhor essa tecnologia que é essencialmente a busca pela fusão do real com o virtual.

Os primeiros projetos de realidade virtual foram desenvolvidos para o uso de simuladores de voo para a Força Aérea dos Estados Unidos (RODRIGUES e PORTO, 2013 apud JACOBSON, 1994). A indústria de entretenimento também foi importante nessa época, pois aplicava a Realidade Virtual no Sensorama, um simulador que possuía uma espécie de cabine que permitia ao usuário expor-se a uma combinação de som estéreo, visão tridimensional, vibrações mecânicas, ar movimentado por ventiladores e até aromas, tudo isso proporcionando ao usuário a participação em uma viagem multissensorial.

Segundo RIBEIRO e ZORZAL (2011), no ano de 1961, os primeiros trabalhos científicos na área foram propostos quando a empresa Philco desenvolveu um par de câmeras remotas e o protótipo de um capacete com monitores que dava ao usuário uma sensação de imersão no ambiente virtual. Mais tarde, esse protótipo passou a se chamar *head-mounted display* (HMD).

A partir desse ponto, várias outras tecnologias foram sendo criadas para melhorar a interação e a imersão do usuário com o mundo virtual como o *Visually Coupled Airbone Systems Simulator* (VCASS), um simulador que usava vídeo-capacetes e computadores interligados para representar o espaço 3D da cabine de um avião, apresentado à Força Aérea Americana por Thomas Furness, em 1982 (RODRIGUES e PORTO, 2013). A *Data Glove* foi uma luva desenvolvida em 1985, capaz de captar a movimentação e inclinação dos dedos da mão (RODRIGUES e PORTO, 2013). O *Oculus Rift* foi uma tecnologia desenvolvida pela empresa Oculus VR, que captou U$2,5 milhões de dólares em uma campanha no Kickstarter em 2012 e veio a ser comprada pela empresa *Facebook* em 2014 (DAVIS; NESBITT; NALIVAIKO, 2015).

Um aplicativo de RV pode apresentar uma simulação de três formas diferentes: passiva, exploratória ou interativa. Uma sessão de RV passiva não apresenta interação do usuário, toda a simulação é controlada pelo *software* e o usuário possui apenas o controle de finalizar a simulação. Uma simulação de RV exploratória permite que o usuário escolha as rotas e pontos

Tekhne e Logos, Botucatu, SP, v.10, n.2, set., 2019.
ISSN 2176 – 4808

de observação no ambiente virtual, porém não apresenta interação com os objetos na cena. Já o tipo interativa permite que o usuário explore e interaja com o ambiente virtual, podendo abrir portas, mover objetos, apertar botões, ou qualquer outro tipo de interação (SILVA; RUSCHEL; OLIVEIRA, 2007).

O objetivo do trabalho foi desenvolver um ambiente em realidade virtual onde o usuário possa interagir em um mergulho marinho para utilização como atração pública em Shoppings.

2 MATERIAL E MÉTODOS

2.1 Equipamentos

Os hardwares utilizados nesse trabalho foram um Processador Intel I5, placa de vídeo GeForce GTX 960, 8Gb de memória Ram, e o Oculus Rift DK2 (*Development Kit 2*), ilustrado na Figura 1.

Figura 1. Oculus Rift DK2

Fonte: Autor, 2019.

As principais especificações do equipamento Oculus Rift DK2 (*Development Kit 2*) são descritos na Tabela 1. Quanto ao *software*, foi utilizada a plataforma de desenvolvimento de jogos Unity 3D, pela sua facilidade de compatibilidade com o Oculus Rift e baixo custo.

Tekhne e Logos, Botucatu, SP, v.10, n.2, set., 2019.
ISSN 2176 – 4808

Tabela 1. Especificações do Oculus Rift DK2

Descrição	**Características**
Exibição	5.7 polegadas OLED (PenTile)
Resolução	1920 x 1080, 960 x 1080 por olho
Taxa de Atualização	75 Hz, 72 Hz, 60 Hz
Persistência	2 ms, 3 ms, full
Campo de Visão	100° (nominal)
Distância Interaxial	63.5mm
Rastreamento	6 graus de liberdade
Rastreamento Rotacional	Giroscópio, Acelerômetro, Magnetômetro
Rastreamento de Posição	Câmera Separada, Sensor Infravermelho próximo CMOS
Taxa de Atualização	Rotational: 1000 Hz; Positional: 60 Hz
Rastreamento Volumétrico	72°H x 52°V (alcance de 8.2 pés)
Latência	~30 ms
Conectividade	USB, HDMI
Peso	440g

A escolha dos hardwares combinadas com técnicas de otimização são de extrema importância no conforto do usuário final, pois se o produto final apresentar mal funcionamento devido a *lags* (intervalo de tempo entre o comando de determinada ação e sua efetivação), pode resultar em um mal-estar para o usuário.

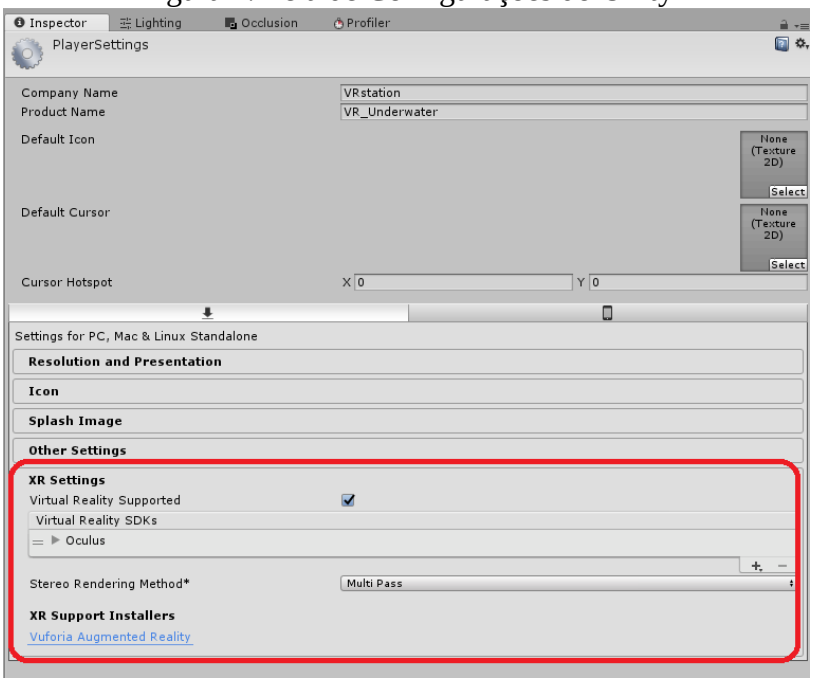
Como a simulação tem restrição de tempo e padronização, foi utilizada uma mescla de conceitos passivo, onde a movimentação é controlada pelo *software* e exploratório, onde a visão é controlada pelo usuário, fazendo o uso de sensores contidos no Oculus Rift para movimentar a câmera virtual, conforme o movimento da cabeça do usuário no mundo real.

2.2 *Softwares*

O Unity já possui a Realidade Virtual integrada no seu *software*, bastando apenas habilitá-la. Dessa forma, em suas configurações (*Player Settings*), foi possível ativar o suporte à realidade virtual (*Virtual Reality Supported*) e escolher o hardware desejado para utilizar, no caso desse projeto, o *Oculus* (FIGURA 2).

Tekhne e Logos, Botucatu, SP, v.10, n.2, set., 2019.
ISSN 2176 – 4808

Figura 2. Tela de Configurações do Unity



Fonte: Autor, 2019.

Nessa simulação, foi utilizado o comando *Recenter* (FIGURA 3) para reposicionar a câmera no ambiente virtual, reiniciando o marco zero do hardware. Ao iniciar a simulação, o *Oculus* se encontra na mão do usuário, abaixo da cabeça e um pouco à frente de seu corpo. Nesse momento, esse ponto é dado como marco zero no mundo real pelos sensores.

Ao colocá-lo na cabeça, a posição da câmera no jogo é alterada conforme as informações passadas aos sensores, resultando em uma posição indesejável como marco inicial. Ao se recentralizar, o Oculus na cabeça no usuário transforma-se o ponto de visão do player como marco inicial sincronizando dessa forma o mundo virtual.

Figura 3. Comando “Recenter” utilizado para iniciar a simulação

```
function Simulate(){
    startTime = Time.time;
    CancelInvoke("OculusTest");
    fadeMaterial.color = fadeColor;
    //Destroy(imgStart);
    imgStart.SetActive(false);
    Cursor.visible = false;
    animalsObj.SetActive(true);
    soundObj.SetActive(true);
    //*** VRstation
    timePass = Time.time;//inicia contagem de tempo
    simStarted = true;
    move=true;
    fader.fadeTime = 2;
    fader.fadeIn = true;
    UnityEngine.XR.InputTracking.Recenter();//recentraliza camera
```

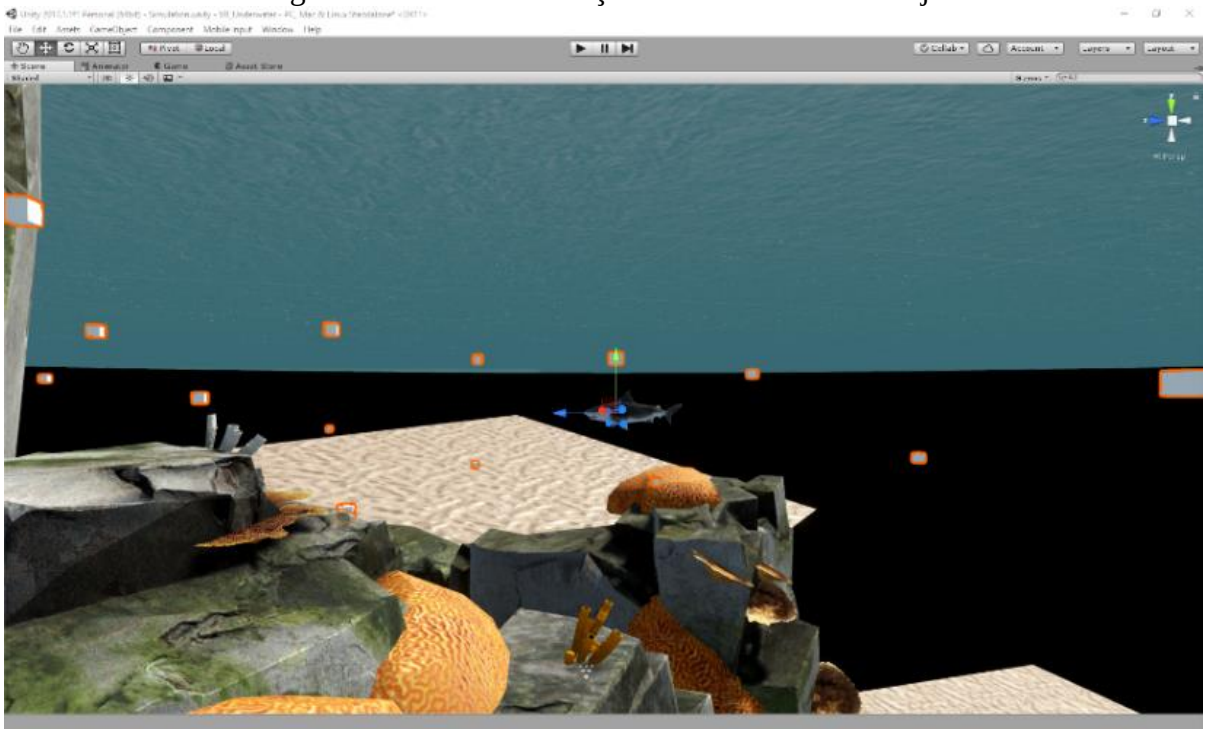
Fonte: Autor, 2019.

Tekhne e Logos, Botucatu, SP, v.10, n.2, set., 2019.
ISSN 2176 – 4808

Foi utilizado o conceito de *checkpoint* para simular o caminho seguido pelos peixes, que fundamentalmente baseia-se em um *script* (trecho de código) que faz com que o objeto se locomova em direção à cubos invisíveis, criando um trajeto em repetição (*looping*) ou uma simples locomoção linear do ponto A ao ponto B. A Figura 4 apresenta os cubos contornados, em laranja, que marcam o caminho específico de um dos tubarões.

Para a simulação dos peixes foram apresentadas animações e variações de velocidade e direção, gerando uma movimentação mais aleatória e natural. Para simular o comportamento dos cardumes, foi desenvolvido o conceito de "*FollowLeader*", onde o primeiro peixe do cardume é definido como líder e os demais seguem seu movimento simulando uma fila. Para deixar essa movimentação menos robótica, a velocidade de movimento dos peixes e a distância entre ele e seu respectivo guia variam de tempo em tempo.

Figura 4. Cubos de marcação contornados em laranja



Fonte: Autor, 2019

A Figura 5 apresenta a função *update* (função nativa do Unity) do *script* que controla a movimentação do cardume.

Tekhne e Logos, Botucatu, SP, v.10, n.2, set., 2019.
ISSN 2176 – 4808

Figura 5. Função update do script "FollowLeader"

```
31
32      // Update is called once per frame
33      void Update () {
34          //Update pro peixe lider
35          if(bigLeader){
36              //Quando chegar na posicao destino atual, sortear uma nova para que mantenha o movimento
37              if(Vector3.Distance(transform.position,targetPos)<10 || targetPos == Vector3.zero){
38                  targetPos = SetTargetPosition();
39                  //sortear nova velocidade de movimento
40                  newSpeed = Random.Range(1,3);
41              }
42              //Trocar velocidade gradualmente
43              speed = Mathf.Lerp(speed,newSpeed,Time.deltaTime);
44              //Movimentar o objeto no sentido Z e na velocidade determinada
45              transform.Translate(0,0,speed*Time.deltaTime);
46              //Calcular nova rotacao do peixe e direciona-lo para o ponto destino
47              Vector3 relativePos = targetPos - transform.position;
48              targetRot = Quaternion.LookRotation(relativePos);
49              transform.rotation = Quaternion.Slerp(transform.rotation, targetRot, Time.deltaTime);
50          }
51          //Update pros demais peixes
52          if(rabbit!=null){
53              //Calcular nova posicao destino
54              targetPos = rabbit.position-rabbit.forward*0.3f;
55              targetPos.y += yVariation;
56              targetPos.x += zVariation;
57              targetRot = rabbit.rotation;
58              //Calcular nova rotacao
59              Vector3 relativePos = targetPos - transform.position;
60              targetRot = Quaternion.LookRotation(relativePos);
61              //Mudar velocidades de movimento e rotacao
62              followSpeed = Mathf.Lerp(followSpeed,newFollowSpeed,Time.deltaTime);
63              rotSpeed = Mathf.Lerp(rotSpeed,newRotSpeed,Time.deltaTime);
64              //Mudar gradualmente para as novas posicoes e rotacoes calculadas
65              transform.position = Vector3.Lerp(transform.position, targetPos, followSpeed*Time.deltaTime);
66              transform.rotation = Quaternion.Slerp(transform.rotation, targetRot, rotSpeed*Time.deltaTime);
67          }
68      }
69
```

Fonte: Autor, 2019.

Outro fator importante analisado foi o ambiente real ao qual o usuário iria utilizar o *software*. Nesse projeto, o foco é utilizá-lo em um quiosque de Shoppings, dessa forma, considerou-se as possíveis ações que os usuários podem executar como gestos com as mãos, quedas, espaço disponível e estrutura, com a finalidade de evitar acidentes com o usuário ou com as pessoas nas proximidades. Assim, a simulação se passa dentro de uma gaiola de mergulho, limitando a movimentação do usuário pelo ambiente.

Nesse projeto, também foi considerada a alocação de peixes em pontos estratégicos, para passarem próximos aos usuários. Apesar da simulação não possuir comandos de gestos, as pessoas tendem a querer tocar nos objetos que aparentam estar ao seu alcance e explorar esse fator pode deixar a experiência ainda mais imersiva.

Como a direção de visão do usuário é livre, não se pode dizer que ele irá aproveitar todos os eventos colocados na simulação, podendo prejudicar sua experiência dependendo da importância do evento. Assim, para os eventos mais importantes, foram adicionados elementos para direcionar a atenção do usuário para certo ponto. Essa técnica foi utilizada na etapa final da simulação, onde o tubarão principal projeta-se contra o lado direito da gaiola. Para aumentar

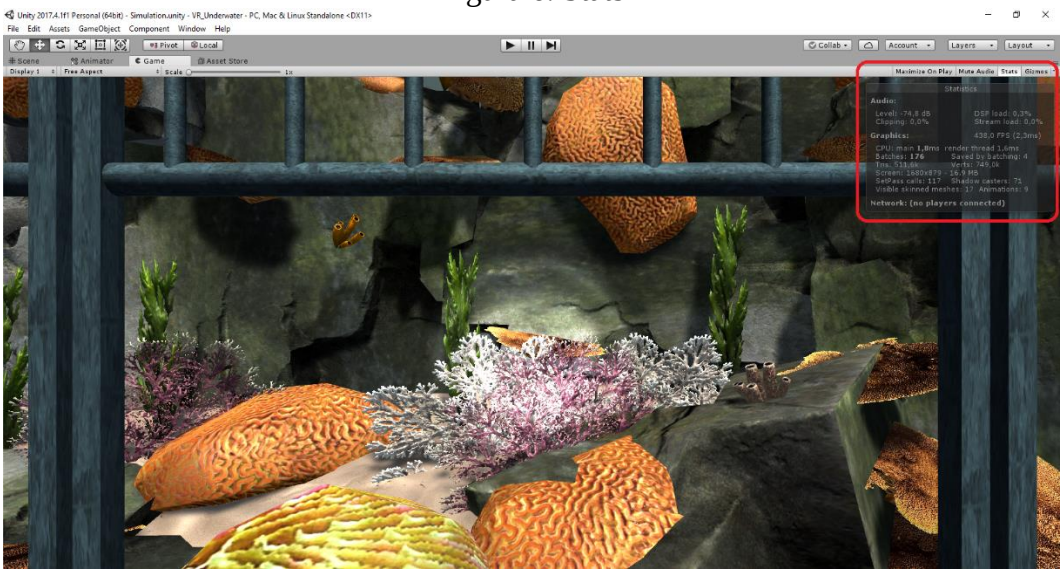
Tekhne e Logos, Botucatu, SP, v.10, n.2, set., 2019.
ISSN 2176 – 4808

a experiência do usuário, foi adicionada uma lula que passa na frente da câmera e continua nadando, chamando a atenção do usuário para onde o evento irá ocorrer.

Essa técnica é muito utilizada em simulações de terror, seja para atrair a atenção do usuário para onde o personagem irá aparecer, ou mesmo para desviar sua atenção de algum ponto e realizar alguma modificação sem que ele perceba.

A RV trabalha com duas telas, uma para cada olho, ou seja, para um hardware, será como processar dois ambientes 3D, o que acaba deixando o aplicativo muito mais "pesado" quando comparado a qualquer outro que não utilize essa tecnologia. Deve-se ter cuidado em relação à quantidade de triângulos e quantidade de Draw Calls que o aplicativo irá gerar. Para visualizar essas quantidades, utilizou-se o "Stats" localizada no canto superior direito da aba "Game" no Untiy, que permitiu habilitar uma janela apresentando informações gráficas da simulação em tempo real. A Figura 6 ilustra a janela habilitada em destaque.

Figura 6. Stats



Fonte: Autor, 2019.

Outros recursos e técnicas dentre os vários disponíveis utilizados na simulação desse projeto foram:

- **Redução de objetos habilitados:** os cenários e animais não são visíveis a todo momento e à toda parte devido a *fog* e as rochas que limitam o campo de visão, assim, os objetos do cenário são habilitados somente quando o usuário ganha visão do local e os demais objetos que saem do campo de visão são desabilitados. Essa técnica reduz a quantidade de materiais usados ao mesmo tempo, minimizando a quantidade de triângulos, uma vez que os polígonos dos objetos desabilitados não são mais contabilizados. Para realizar esse processo, utilizou-se

Tekhne e Logos, Botucatu, SP, v.10, n.2, set., 2019.
ISSN 2176 – 4808

colisores marcados como *triggers* que são ativados quando o usuário passa por eles, dando a posição do usuário no ambiente virtual 3 identificando os objetos que ele não terá visão.

- Oclusion Culling: utilizou-se um recurso do Unity que contabiliza apenas o que está visível na tela, excluindo a contagem de triângulos de todos os objetos que a câmera não estiver renderizando. Pela aba *Window* foi selecionado o *Oclussion Cullling* no menu *Drop Down*. Pela configuração *Bake* foi possível calcular as áreas que serão habilitadas conforme entram na visão da câmera.

- Atlas de texturas: cada textura utilizou um material para ser aplicada em um objeto, sendo assim, quanto mais texturas mais materiais, o que significa um aumento no número de *Draw Calls* na sua simulação. Foi criado um atlas de texturas que reuniu várias texturas pequenas em apenas uma grande, reduzindo drasticamente o número de materiais utilizados. Existem diversos *softwares* que geram um Atlas e nessa simulação foi utilizado um *asset* do Unity chamado *Mesh Baker*.

- Lightmap: é responsável por estampar a iluminação em todos objetos marcados como estático para evitar o processamento de sombra dinâmica sobre os mesmos. As sombras são indispensáveis para que se possa obter uma realidade convincente, porém aumentam significativamente o número de polígonos processados. Esse recurso foi utilizado acessando a janela *Lighting* do Unity.

Redução de polígonos dos objetos: no caso de modelos 3D, existem muito detalhados que podem "carregar" muito a simulação e alguns *assets* podem realizar uma simplificação. Esse processo utilizou um *asset* chamado *Mesh Simplify* que gerou uma certa deformação dos objetos ao reduzir sua quantidade de triângulos. De forma equilibrada, o uso da técnica pode gerar resultados satisfatórios com a melhor performance de seu aplicativo.

3 RESULTADOS E DISCUSSÃO

Para que o usuário não tenha uma mudança repentina do mundo real para o ambiente virtualizado, a simulação inicia-se com um painel preto, impedindo totalmente sua visão do mundo virtual para posteriormente executar o comando que sincroniza a sua direção com a da câmera no ambiente virtualizado. Junto a esse comando, retira-se gradualmente o painel preto da tela, deixando visível todo o ambiente virtualizado para o início da simulação.

No resultado final, observou-se que o uso da técnica de iluminação *Directional Light*, que tem a função de simular o Sol, sua cor e sua intensidade, afetou muito como os objetos

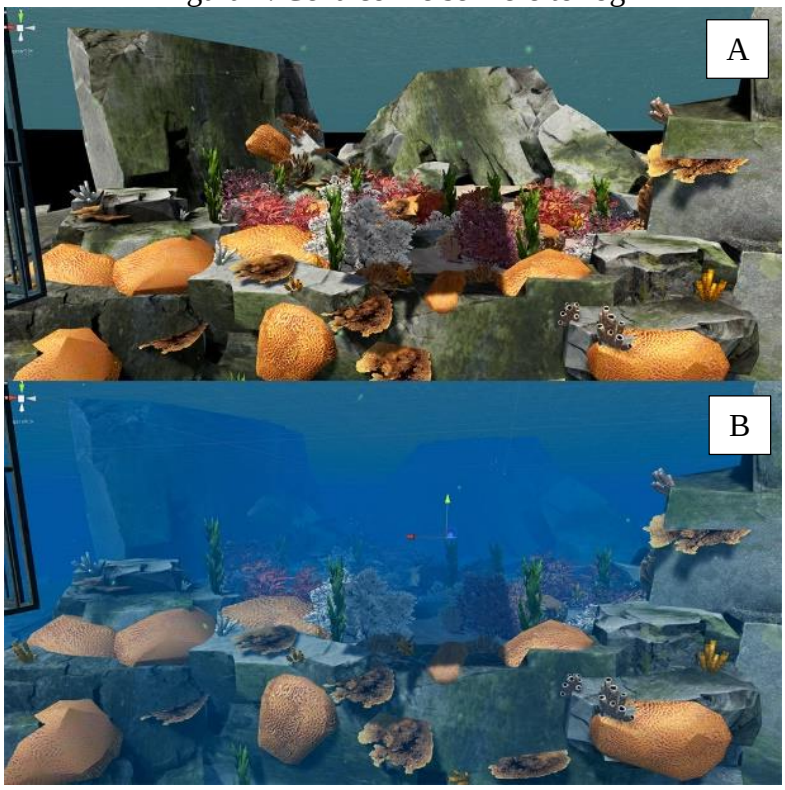
Tekhne e Logos, Botucatu, SP, v.10, n.2, set., 2019.
ISSN 2176 – 4808

surgiram nos cenários. Nesse efeito, o desenvolvedor deve atentar-se ao uso excessivo de intensidade para que a cor dos demais objetos não seja afetada.

Nos efeitos de sombras, verificou-se uma aproximação do ambiente virtualizado ao mundo real onde períodos de noite e dia apresentam variações nas sombras sob os objetos. A percepção desses efeitos com uso correto é de vital importância para um resultado realista, porém, deve-se balancear seu uso para que não exista sobrecarga no processamento de hardware, pois o efeito sombra pode dobrar a quantidade de triângulos processados nos objetos 3D.

O efeito *fog* utilizado para poder criar o embaçamento gradual dos objetos distantes melhorou bastante a ambientação pois quanto mais afastado um objeto ficou do observador, sua cor azul se tornou mais intensa até o momento do seu desaparecimento total. Esse efeito é essencial para criar uma ambientação submersa proposta nesse projeto. A Figura 7 demonstra a diferença da imagem sem o efeito (A) e com o efeito (B).

Figura 7. Cena com e sem efeito fog



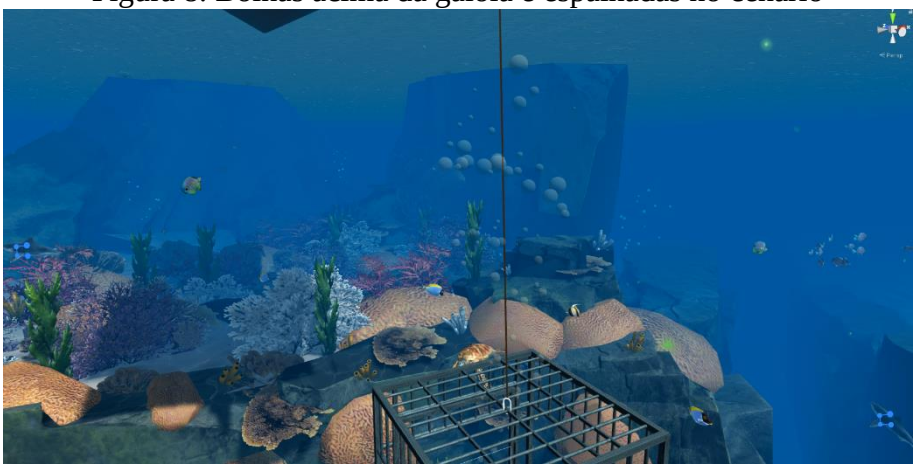
Fonte: Autor, 2019.

Para intensificar o nível de detalhamento do simulador, o efeito de bolhas utilizado permitiu que a simulação ficasse muito mais convincente criando um dinamismo adicional e não apresentando somente um ambiente com uma coloração mais azulada. Na Figura 8, nota-

Tekhne e Logos, Botucatu, SP, v.10, n.2, set., 2019.
ISSN 2176 – 4808

se as bolhas originadas junto ao áudio de expiração do mergulhador e algumas espalhadas pelo coral.

Figura 8. Bolhas acima da gaiola e espalhadas no cenário



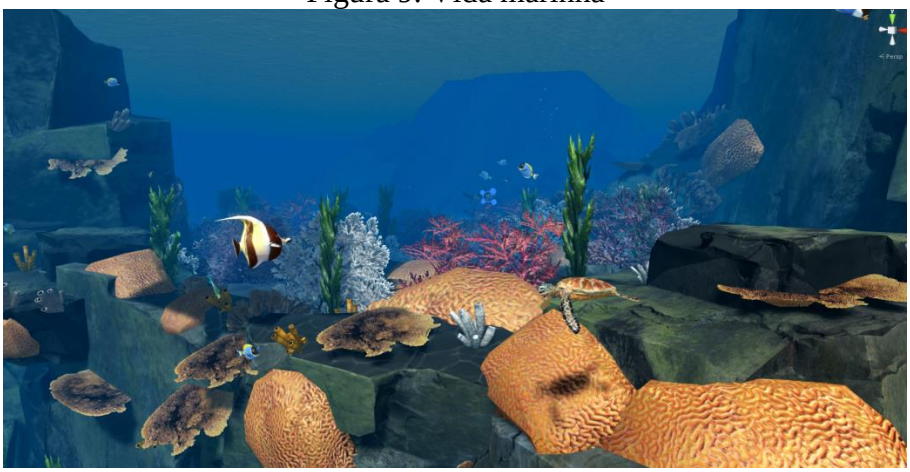
Fonte: Autor, 2019.

Notou-se que o uso do som teve papel fundamental na imersão do usuário no simulador assim como tem vital importância em outras mídias, como filmes. Para essa simulação, foram utilizadas duas trilhas sonoras para auxiliar na sensação desejada, uma mais calma na primeira parte, com uma sensação de relaxamento e contemplação do ambiente e uma mais tensa, com o intuito de passar sensação de perigo, quando o usuário chegar na cena do tubarão surgindo em sua direção.

Foram utilizados sons que não são realistas de um ambiente aquático, mas propositalmente utilizados para atingir sensações desejadas em determinados momentos da simulação. Os demais efeitos sonoros foram utilizados o mais próximo possível de sons reais, comoo indivíduo interagindo como a própria respiração através do equipamento de mergulho, som de bolhas exaladas, sons da vida nos corais que se assemelham a cliques, entre outros. Corais, peixes crustáceos e outros demais modelos de animais marinhos adicionados às cenas proporcionaram uma exploração intensa e diversificada, assim como encontrada em ambientes reais (FIGURA 9).

Tekhne e Logos, Botucatu, SP, v.10, n.2, set., 2019.
ISSN 2176 – 4808

Figura 9. Vida marinha



Fonte: Autor, 2019.

A Realidade Virtual pode gerar enjoos nos usuários, pois o movimento detectado pela visão difere-se dos sinais esperados pelo sentido vestibular. Durante o desenvolvimento, a simulação apresentou conflitos de sinais em dois casos. No primeiro, quando o usuário se movimentou no ambiente virtual, pois o corpo recebeu estímulos visuais de movimentos ao mesmo tempo que se encontrava parado fisicamente, causando uma contradição entre os estímulos. Para minimizar esses efeitos, é necessário evitar movimentos muito bruscos durante o desenvolvimento.

O segundo caso de conflito ocorreu quando os usuários executaram movimento reais, como a rotação da cabeça. Esses movimentos devem ser respondidos com o mínimo de latência possível na simulação para que não se perca a sincronização com o movimento real. Assim, se o simulador apresentar muito *lag*, esse atraso da resposta de movimento pode ocasionar enjoos aso usuários.

4 CONCLUSÃO

O aplicativo obteve o resultado desejado, onde os usuários realizaram testes e utilizaram o ambiente virtual sem apresentar sintomas de enjoos. Observou-se também diferentes reações nas etapas de submersão, contemplação do ambiente subaquático e dos objetos se movimentando, como o ataque de tubarões.

Com a utilização das técnicas de otimização, foi possível reduzir o número de triângulos dos *meshes* em cena, possibilitando a adição de mais objetos e elaborar uma cena mais complexa e preenchida sem que prejudicasse o processamento do equipamento.

Tekhne e Logos, Botucatu, SP, v.10, n.2, set., 2019.
ISSN 2176 – 4808

O projeto foi aplicado em todas as estações de empresa de Realidade Virtual e Aumentada de Botucatu-SP e utilizada em diversos shoppings no Brasil.

REFERÊNCIAS

DAVIS S.; NESBITT K.; NALIVAIKO E.; **Comparing the onset of cybersickness using the Oculus Rift and two virtual roller coasters**. In: Australasian Conference on Interactive Entertainment, 11., 2015, Sydney, Austrália. Disponível em: <http://crpit.com/confpapers/CRPITV167Davis.pdf> Acesso em: 18 out. 2018.

KIRNER C.; TORI R. **Realidade Virtual Conceitos e Tendências**. Pré-Simpósio VII Symposium on Virtual Reality, 2004. Disponível em: <http://ckirner.com/download/capitulos/livro_pre_simp-2004.pdf#page=12>. Acesso em 18 out. 2018.

RIBEIRO M. W. S; ZORZAL E. R.; **Realidade Virtual e Aumentada: Aplicações e Tendências**. In: Simpósio de Realidade Virtual e Aumentada, 13., 2011, Uberlândia, MG.

RODRIGUES, G. P.; PORTO C. M. **Realidade Virtual: Conceitos, Evolução, Dispositivos e Aplicações**. Interfaces Científicas, Aracaju, v.1, n.3, p. 97-109, jun. 2013. Disponível em: <https://periodicos.set.edu.br/index.php/educacao/article/view/909/414>

SILVA T. B.; RUSCHEL R. C.; OLIVEIRA A. A.; **Avaliação do projeto com realidade virtual e sua Avaliação pós-ocupação: uma comparação**. In: Encontro de tecnologia de informação na construção civil, 3., 2007, Porto Alegre. Disponível em: <http://noriegec.cpgec.ufrgs.br/tic2007/artigos/A1090.pdf>.

UNITY DOCUMENTATION. **ScriptReference**. Disponível em: <https://docs.unity3d.com/ScriptReference/>. Acesso em: 30 set. 2018.

UNITY DOCUMENTATION. **VR overview**. Disponível em: <https://docs.unity3d.com/Manual/VROverview.html>. Acesso em: 30 set. 2018.

UNITY DOCUMENTATION. **Optimisation for VR in Unity**. Disponível em: <https://unity3d.com/pt/learn/tutorials/topics/virtual-reality/optimisation-vr-unity>. Acesso em: 30 set. 2018.