Sistema Multijogador com NodeJS
Uma aplicação móvel de realidade aumentada utilizando NodeJS para interações multijogador

Gustavo Reis, Sandro Laudares

PUC Minas em Betim
Bacharelado em Sistemas de Informação

Gustavo.caetano@outlook.com.br, laudares@pucminas.br

Este trabalho visa criar um sistema multijogador para jogos na plataforma móvel, ou seja, múltiplos jogadores em um mesmo ambiente 3D em realidade aumentada. O alto custo de hardware e licenças de terceiros para a utilização de interações multijogador tem levado à busca de soluções mais eficientes e econômicas. Ao final, foi desenvolvido um sistema utilizando o interpretador de código aberto JavaScript NodeJS no lado servidor e no lado cliente, o ambiente 3D feito na engine Unity3D.

1. Introdução

O crescente mercado de jogos para computador, consoles, smartphones, e o surgimento de novas mecânicas de jogos (multijogadores) tem crescido e popularizado a utilização de tecnologias que possibilitam a comunicação cliente-servidor de forma eficiente e com menor custo. A disponibilidade de ambientes de desenvolvimento que possibilitam a exportação do código fonte para várias plataformas têm facilitado o desenvolvimento de jogos para uma ampla variedade de dispositivos. Com isso, se destaca o ambiente de desenvolvido da Unity3D, pois além de prover uma licença pessoal gratuita para usuários que não faturam mais que $100.000 dólares, dispõe de uma ampla variedade de dispositivos compatíveis para exportação.

A quantidade de jogadores em um servidor varia de 2 a 100 jogadores simultâneos como nos jogos “Ark Survival Evolved” e “Call of Duty”. Isso é um limite imposto pela capacidade de tratar várias requisições simultâneas no mesmo servidor. Além disso, engines limitam ou cobram uma taxa para a utilização do recurso multijogador. A Unity3D, por exemplo, limita em 8 usuários simultâneos em um jogo multijogador utilizando a licença gratuita. A utilização de tecnologias escaláveis que utilizam “melhor” os recursos no servidor bem como disponibilizam um acesso fácil para desenvolvedores com pouco recurso financeiro, abrem a possibilidade de aumentar a quantidade de jogadores, diminuindo o custo de aquisições de vários servidores juntamente com licenças e aumento da quantidade de interações em jogos multijogador em tempo real.

As linguagens de programação como Java, PHP e C, trabalham com a criação de um thread para cada requisição e consecutivamente aumenta a utilização de recursos da memória física do servidor fazendo com que diminua a quantidade de conexões simultâneas suportadas. A maioria das linguagens tem comportamento bloqueante no

thread em que estão, ou seja, se um cliente faz uma consulta pesada no banco de dados, a thread utilizada fica travada até essa consulta terminar.

Com a necessidade de criar uma solução que gerencia melhor múltiplas conexões com o servidor, foi desenvolvido o interpretador em JavaScript NodeJS que funciona no lado do servidor. O NodeJS é fundamentado no interpretador de código aberto criado pela Google V8 JavaScript Engine e baseado em C++ que funciona no navegador Chrome; foi criado por Ryan Dahl em 2009 e seu desenvolvimento é mantido pela empresa Joyent. Por ter licença de código aberto, possibilita uma grande economia no requisito licença.

NodeJS é um recurso excelente para interações ao vivo em um site ou programa que possui muitas solicitações de entrada e retorno de dados, pois é orientado a eventos não síncrono e não bloqueante ininterruptamente beneficiando as requisições de dados de entrada e saída no servidor. Nisso, sua utilização em um servidor de jogos pode aumentar o desempenho e um número maior de requisições em um único servidor.

Este texto está estruturado em 5 seções. A seção 2 apresenta o referencial teórico do projeto proposto, e são apresentadas referências de vários autores no campo de estudo mais abrangente para o mais específico, são eles: Sistemas Distribuídos, Realidade Aumentada e NodeJS. A seção 3 descreve os projetos de terceiros criados ou em desenvolvimento para sistemas interativos multijogador. Na seção 4, temos uma breve informação da metodologia escolhida e, na seção 5, foi discutido como o desenvolvido do sistema foi executado. E, por fim, temos na conclusão, o que foi possível e não possível realizarmos.

2. Referencial Teórico

2.1. Sistemas Distribuídos

Um sistema distribuído é aquele cuja definição se resume como um conjunto de unidades ou nós independentes. Estas unidades independentes, se comunicam processando a aplicação em diferentes localidades fazendo com que o usuário tenha a impressão de que toda a aplicação é gerenciada por um sistema único.

Na aplicação dos sistemas distribuídos, a utilização da programação orientada a objeto é de grande importância, pois traz vantagens no uso da abstração de dados e encapsulamento de objetos. Um sistema distribuído deve ser visível para o usuário como um único sistema, abstraindo todo o conjunto de unidades. “Os objetos distribuídos são importantes para trazer as vantagens do encapsulamento e da abstração de dados para os sistemas distribuídos e também as ferramentas e técnicas associadas do campo do projeto orientado a objetos. (COULOURIS, 2013).

A programação orientada a objetos também possibilitou a reutilização de aplicações e módulos diferentes e distribuídos em unidades distintas trocando informações para se comportarem como um único sistema. Num sistema distribuído é possível coexistir aplicações diferentes que se comunicam tendo a possibilidade da reutilização destas aplicações. “Vista como uma técnica para aumentar a produtividade e baixar os custos, a reutilização de software é um dos principais estudos dos pesquisadores da área de computação. ” (ROSSI, 2009)

Um dos problemas enfrentados pelas aplicações distribuídas é a dificuldade na comunicação entre aplicações diferentes e em regiões ou localidades distintas. O ambiente a World Wide Web reflete bem esta característica. A comunicação destes sistemas deve ser transparente entre eles, possibilitando a troca eficiente das informações. “A popularização da Internet criou expectativas e promessas quanto ao uso mais inteligente dos dados disponíveis, culminando na necessidade de interoperabilidade entre aplicações”. (DAMASCENO, 2005)

2.2. Realidade Aumentada

Realidade aumentada é a técnica de interação de objetos virtuais no mundo real. Isto é possível através do processamento de imagens e reconhecimento de padrões realizados pelo software. Após a captura das imagens ou vídeos pelo dispositivo eletrônico, os dados são processados e padrões são reconhecidos, possibilitando a inserção de objetos virtuais na perspectiva das imagens ou vídeos do mundo real.

“As habilidades espaciais, em particular a de visualização, são intensamente requeridas por inúmeras profissões artísticas, técnicas e científicas ”. (SEABRA, 2009). A manipulação de vídeos e imagens com o objetivo de modificar a realidade, fazem parte do escopo das habilidades espaciais que são estudadas e aplicadas por várias áreas do conhecimento artístico, técnico e científico.

A aplicação da habilidade espacial de realidade aumentada, pode ser utilizada em várias áreas de estudo como geografia, biologia, artes, matemática (geometria), entre muitas outras aplicações. “Aplicações deste tipo apresentam características interessantes para áreas de aplicação como: sensoriamento, monitoramento, visualização e interação em ambientes inteligentes e ubíquos”. (TORRES, 2008)

A aplicação da realidade aumentada também pode servir como ferramenta de treinamento para usuários com problemas cognitivos através da criação de jogos interativos criados para a educação destes. No livro “Realidade Aumentada e Gamificação[1] A tecnologia no auxílio de alunos com deficiência intelectual” diz: “Este livro tem como objetivo apresentar uma discussão sobre o uso da tecnologia de realidade aumentada e gamificação[1] aplicadas a educação. Para isso, foi desenvolvido o aplicativo AR+G Atividades Educacionais, com o intuito de auxiliar no processo de aprendizagem de alunos com deficiência intelectual. Os resultados apontaram que o aplicativo contribui para a melhoria de habilidades que, pelos métodos tradicionais de ensino utilizados até o momento pelos alunos, ainda não haviam sido melhoradas”. (COLPANI, 2015).

2.3. NodeJS

NodeJS é um interpretador em Java script que funciona no lado servidor. Ele tem como objetivo a criação de aplicações de alta escalabilidade (como um servidor web), com

[1]Gamificação: é o uso de mecânicas e dinâmicas de jogos para engajar pessoas, resolver problemas e melhorar o aprendizado, motivando ações e comportamentos em ambientes fora do contexto dos jogos (ESPÍNDOLA, 2016)

códigos capazes de manipular dezenas de milhares de conexões simultâneas, numa máquina física.

Com o crescimento dos chamados “Web Services” (sistemas de interação e comunicação na web), a quantidade de diferentes sistemas dificultou a comunicação e a troca de dados entre eles. Por isso, foi proposto o chamado APIs RESTful, que na verdade é uma forma arquitetural que veio para simplificar a comunicação entre diferentes sistemas e protocolos. “Desde 2010, o NodeJS provou ser uma excelente plataforma escalável na solução de diversos problemas, principalmente para construção de APIs RESTful”. (PEREIRA, 2016).

Ter uma aplicação escalável é possuir a capacidade de execução e disponibilidade desta, mesmo com momentos de alta carga. “O NodeJS é uma poderosa plataforma para construir de forma rápida e fácil aplicações de rede escaláveis. Utiliza um modelo de single thread, faz I/O não bloqueante e por isso trabalha sempre de forma assíncrona. ” (MORAES, 2015).

A linguagem de programação JavaScript é utilizada habitualmente como programação no lado do cliente em arquitetura cliente-servidor. Mas NodeJS traz uma mudança de paradigma, pois utiliza a programação JavaScript também no lado servidor. Isso, facilita o desenvolvimento de aplicações web pois tira proveito da familiaridade da sintaxe do JavaScript pelos desenvolvedores. “NodeJS é uma poderosa plataforma. Ele permite escrever aplicações JavaScript no server-side, tirando proveito da sintaxe e familiaridade da linguagem para escrever aplicações web escaláveis”. (PEREIRA, 2014)

3. Trabalhos Relacionados

Lance (2017) é um servidor de jogos baseado em NodeJS. Criado para permitir que desenvolvedores criem jogos online multijogador sem terem preocupação com a implementação do código de sincronização de rede. Ele se esforça para proporcionar experiência tranquila para o desenvolvedor e jogador mesmo ocorrendo o atraso de dados no sincronismo em rede.

Colyseus (2017) é um servidor de jogos NodeJS baseado em Websocket. Ele é compatível com os motores de jogos Unity3D, Construct 2 entre outros recursos de compatibilidade. Utiliza a licença MIT que dá o direito para usar, copiar, modificar, unir com outros softwares, publicar, distribuir.

Gordon (2013) é um servidor leve para o desenvolvimento de aplicativos multiusuários e jogos com HTML5 e Adobe Flash / Air. Ele utiliza o NodeJS para processamento dos dados dos clientes.

Uma característica comum das aplicações descritas acima é a utilização do NodeJS como componente de rede (Netcode). Netcode é o código de rede responsável pelo gerenciamento e sincronismo dos dados entre o servidor e os clientes. As aplicações Lance, Colyseus e Gordon são licenciadas no padrão código livre e dispõem de documentação pública na web, além de uma comunidade de desenvolvedores para troca de informações e conhecimento.

4. Metodologia

Este trabalho apresenta o desenvolvimento de um sistema multijogador para jogos usando os recursos do interpretador de código JavaScript NodeJS. A criação e teste da aplicação em ambiente fechado foi a metodologia utilizada para viabilizar este trabalho. As etapas desenvolvidas foram as seguintes:

4.1. Etapas do trabalho

As etapas do trabalho foram divididas conforme abaixo.

a) Levantamento de requisitos:

Os jogos de realidade aumentada vêm crescendo no ambiente dos jogos digitais, sobretudo, nos dispositivos móveis. Foi requisito a criação de um ambiente 3D de realidade aumentada com características multijogador em dispositivos móveis.

Para o sistema multijogador, foi analisado a ferramenta de desenvolvimento Unity3D. A engine Unity3D oferece um módulo gratuito para até 8 jogadores simultâneos em um jogo multijogador. Este limite, somente deve ser ultrapassado caso seja contratado uma licença paga. Foi avaliado que a função multijogador também deveria estar presente em nosso simulador 3D de realidade aumentada sem contratempo imposto da versão gratuita da engine Unit3D.

O sistema foi construído sobre a engine de desenvolvimento Unity3D. A linguagem utilizada no cliente foi a CSharp, e o compilador foi o MonoDevelop (padrão da Unity3D). No lado servidor, foi utilizado o interpretador de JavaScript NodeJS.

b) Modelagem do sistema:

A arquitetura do sistema é baseada em cliente-servidor. Sendo o servidor responsável pela transmissão dos dados entre os clientes.

c)Desenvolvimento do sistema:

No desenvolvimento do sistema, foram apresentadas as interações dos módulos cliente/servidor e as etapas de construção do jogo.

d) Testes:

O sistema foi testado através de clientes virtuais em um computador. E em dispositivos móveis utilizando o sistema operacional Android.

4.2. Ambiente de Desenvolvimento

O ambiente de desenvolvimento é composto de uma estação de trabalho com o sistema operacional Windows 10 e o programa de desenvolvimento Unity3D corretamente instalado com o plug-in Vuforia. O servidor é composto pela instalação do NodeJS, bem como o módulo Express, em uma máquina com sistema operacional Ubuntu server.

Neste trabalho, o ambiente é composto pelas seguintes ferramentas apresentadas na Tabela 1.

Tabela 1. Ferramentas do ambiente de desenvolvimento.

Funcionalidade	Ferramenta
Ambiente 3D	Unity 3D 5.5.0
Realidade aumentada	Plug-in Vuforia
Programação do servidor	NodeJS 8.4.0
Sistema Operacional	Windows 10, Android 4.4 e Ubuntu Server 14
Plug-in conexão com servidor	Socket.io
Módulo interpretador web	NodeJS Express

Na ferramenta de desenvolvimento Unity3D, foram instalados os plug-ins Vuforia para criação da interface de realidade aumentada e o plug-in Socket.io para comunicação com o servidor. No lado servidor, foi instalado o NodeJS integrado com o plug-in Express.

5. Desenvolvimento

5.1. Levantamento de Requisitos

Escolhemos a Unity3D para a plataforma de desenvolvimento 3D, pois oferece uma versão gratuita que é possível construir jogos com todos os recursos proposto no levantamento de requisitos, desde que o faturamento não seja superior a $ 100.000 dólares; assim, facilita o início para pequenos desenvolvedores de jogos. E também é possível exportar para várias plataformas clientes como, por exemplo, televisores digitais, computadores de mesa, dispositivos móveis, sistemas operacionais (Windows, Android entre outros).

Atualmente, os dispositivos móveis são “populares” mundialmente e a maioria dos dispositivos possuem recursos físicos capazes de executar um jogo 3D. Com isso, escolhemos exportar o jogo para a plataforma de celular do sistema operacional Android.

Plug-ins e módulos: Além do módulo socket.io para conexão do cliente ao NodeJS, foi utilizado o módulo Express no lado do servidor NodeJS. Express é um framework para aplicações web e móvel fornecendo um conjunto robusto de recursos para o servidor.

5.2. Modelagem do Sistema

A figura 1 mostra as interações de recebimento e envio de posicionamento dos jogadores 1 e 2. Ambos fazem uso do plug-in socket.io no lado do cliente a qual envia e recebe os dados de posicionamento do servidor rodando o NodeJS.

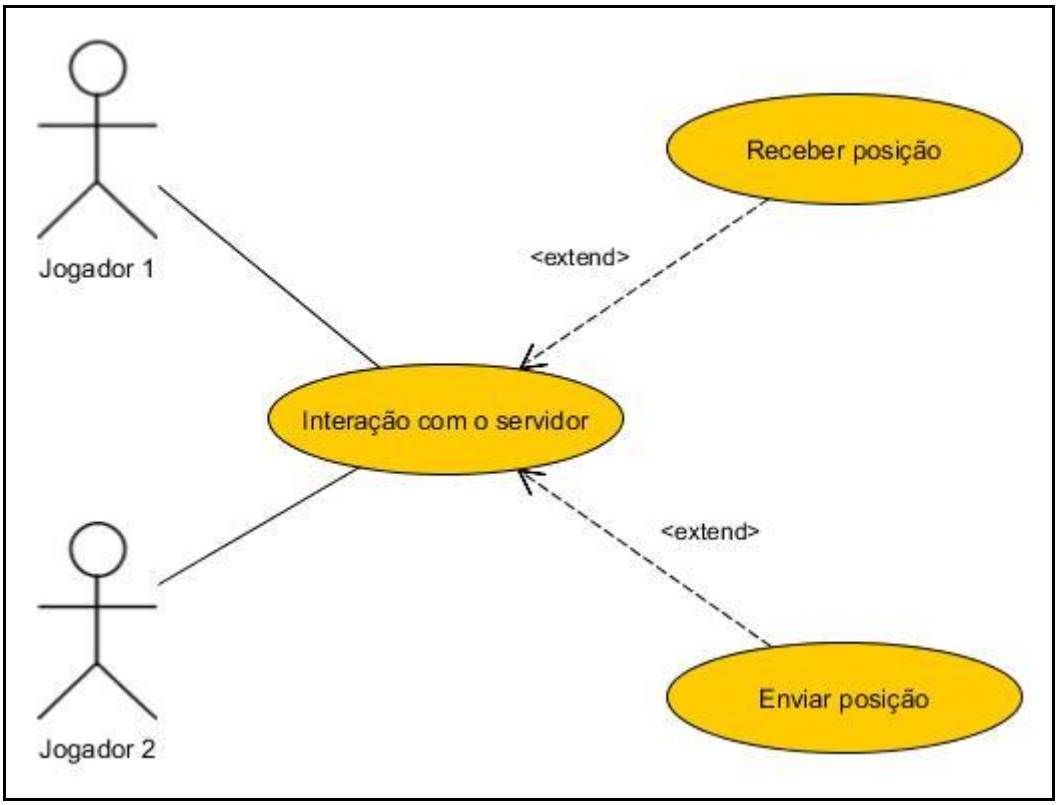

Figura 1. Diagrama de caso de uso onde mostram as interações cliente-servidor do sistema multijogador.

5.3. Interface do Sistema

A interface do sistema foi projetada em realidade aumentada utilizando os recursos dos componentes físicos do celular, a saber, câmera para captura das imagens e tela touch para interações no ambiente 3D, além, do hardware necessário para executar a aplicação. Ao utilizar os recursos do plug-in Vuforia para Unity3D, foi possível criar um ambiente 3D interativo onde foi necessário a digitalização de uma imagem a ser reconhecida pelo plug-in da Vuforia. Esta imagem é tomada como referência para construção do ambiente 3D (Figura 02 e 03).

Figura 2. Imagem impressa em uma folha do tamanho A4 utilizada como referência para construção do ambiente 3D virtual pelo plug-in Vuforia.

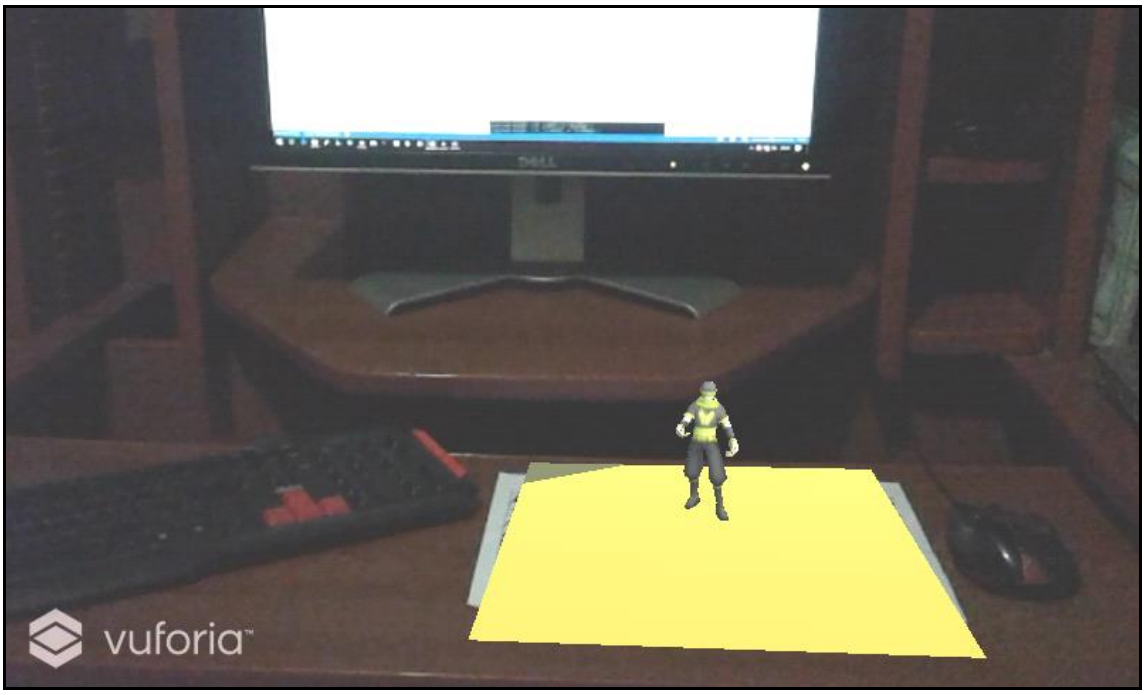

Figura 3. Mostra o ambiente 3D gerado sobre a imagem real impressa numa folha de tamanho A4.

5.4. Programação

O sistema foi composto no lado cliente pelas seguintes classes: ScreenClicker, Network, NetworkMove, NavigatePosition, ClickMove.

ScreenClicker é a classe responsável por reconhecer as interações do touch sobre a área previamente demarcada no ambiente 3D. Após um toque na tela touch, é verificado se a posição faz parte da área previamente escolhida. Se sim, é acionado a classe ClickMove que transmite o comando para o personagem através da classe NavigatePosition e retransmite para a classe NetworkMove. A classe NetworkMove envia os dados de cada personagem para o servidor e o servidor retransmite para todos os clientes utilizando a classe Network.

A figura 4, demostra o funcionamento do nó servidor, retransmitido as informações de posicionamento de cada jogador.

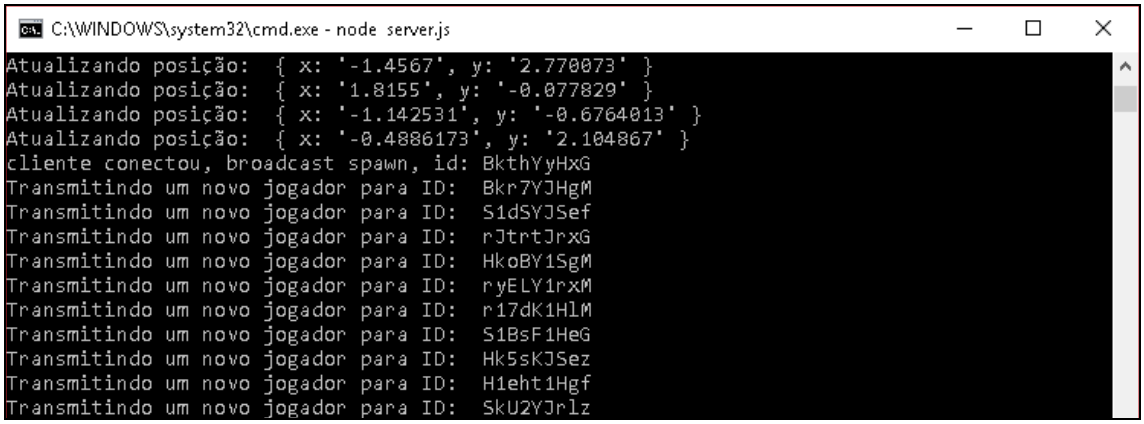

Figura 4. Console do servidor – Interações transmitidas pelo servidor utilizando o NodeJS.

5.5. Testes

Foram realizados dois tipos de teste: em um ambiente virtual único e outro utilizando os dispositivos móveis.

O ambiente virtual único é composto por um único computador emulando vários dispositivos móveis que executam a aplicação. Com isso, foi possível testar o servidor NodeJS com vários clientes conforme mostra a figura 05. Foram feitos paralelamente 11 clientes. Este número foi limitado pela capacidade da placa de vídeo no computador, pois a placa chegou ao limite dos recursos.

Já o ambiente de teste com dispositivos móveis foi testado somente com dois clientes conectados ao servidor.

Para o teste nos dispositivos móveis, foram utilizados smartphones com sistema operacional Android superior à versão 4.4. O recurso multijogador funcionou com boa performance e não houve degradação no servidor, ou seja, a diferença do tempo de resposta não foi perceptível ao ser humano.

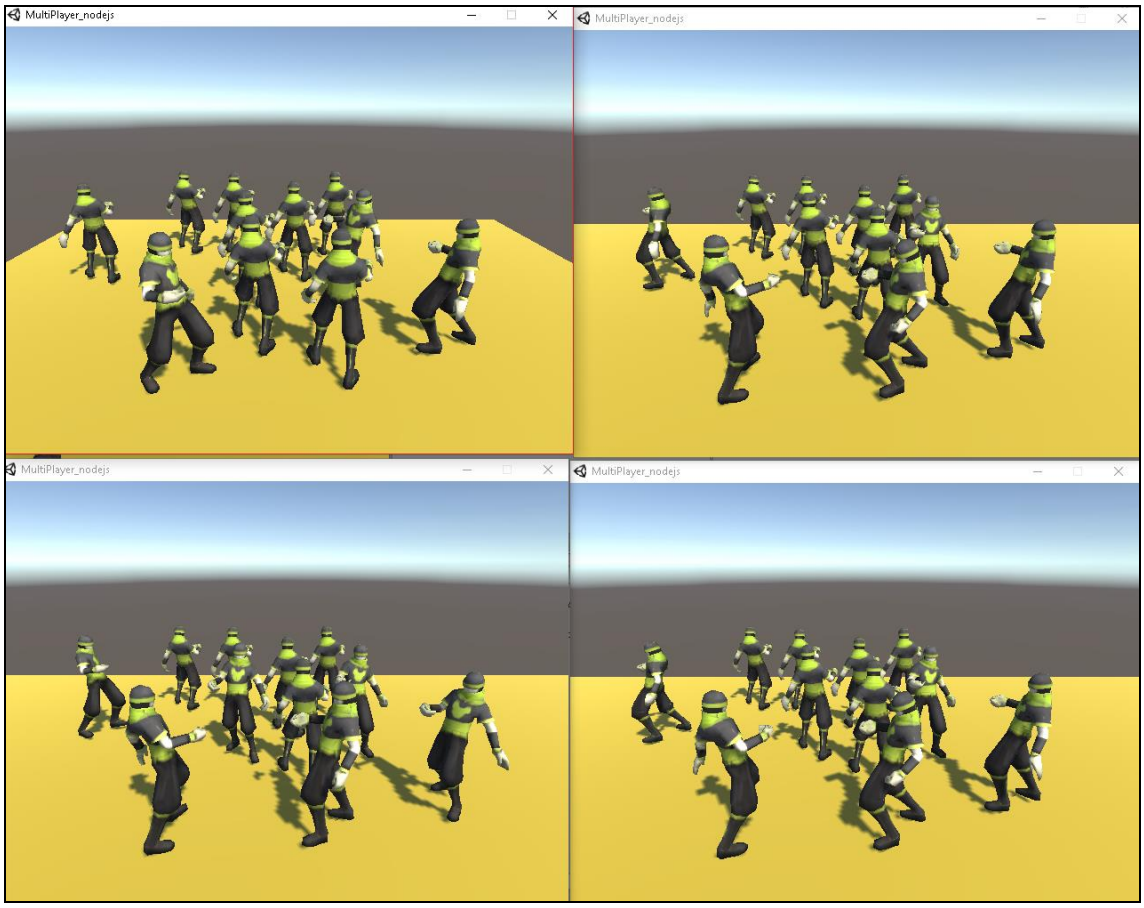

Figura 5. Vários clientes conectados em um ambiente multijogador.

6. Conclusão

Neste trabalho, foi desenvolvido um sistema multijogador para dispositivos móveis utilizando o recurso de realidade aumentada do plug-in Vuforia. A plataforma de desenvolvimento Unity3D correspondeu ao objetivo de criação do ambiente 3D e integração do plug-in de comunicação multijogador socket.io junto ao NodeJS. Os testes em ambiente fechado foram realizados com sucesso atingindo o objetivo proposto.

Como o teste foi realizado em um ambiente fechado e controlado, ou seja, foi testado levando em consideração somente dois dispositivos móveis e também foi realizado somente por um desenvolvedor, novos resultados podem surgirem com testes de estresse e inclusive a possibilidade de um teste em ambiente aberto (para o público em geral) utilizando a loja virtual Google Play para distribuição ao público.

A principal contribuição desse projeto foi a implementação de um recurso cada vez mais utilizado pelos jogos de computador e móvel: a interação multijogador. Recursos em hardware de computador pode serem otimizados e licenças de software pagos podem serem substituídos por esta solução de código livre[1].

Portanto, em futuros trabalhos pretende-se criar outras interações multijogador além do posicionamento utilizando os recursos de extrapolação[2] e interpolação[3] de interações multijogador disponibilizado pelo sistema Lance.

---

[1] Código livre: é um modelo de desenvolvimento que promove um licenciamento livre para o design ou esquematização de um produto e a redistribuição universal desse design ou esquema, dando a possibilidade para que qualquer um consulte, examine ou modifique o produto.

[2] Extrapolação: É um método de sincronização entre cliente e servidor. Na ausência de dados de sincronização do servidor, devido a algum fator de falha, é possível prever sequência de interações até certo ponto no lado cliente.

[3] Interpolação: É um método de sincronização entre cliente e servidor. Nesse método, os clientes executam etapas que anteriormente eram enviadas pelo servidor utilizando de cálculos.

Referências Bibliográficas

PEREIRA, Caio Ribeiro. **Aplicações web real-time com NodeJS.** Editora: Casa do Código, 2014.

PEREIRA, Caio Ribeiro. **Construindo APIs REST com NodeJS.** Editora: Casa do Código, 2016.

COLYSEUS. **Multiplayer Game Server for NodeJS.** <https://github.com/gamestdio/colyseus>. Acesso em: 20 de novembro 2017.

DAMASCENO, Luciano Lança. **Interoperabilidade de metadados em aplicações distribuídas: desenvolvimento de ferramentas para validação de metamodelos**. Biblioteca Digital da Unicamp, 2005.

ESPÍNDOLA, Rafaela. **O que é a gamificação e como ela funciona?** <https://www.edools.com/o-que-e-gamificacao/>. Acesso em 8 de dezembro.

COULOURIS, George; BLAIR, Gordon; DOLLIMORE, Jean; KINDBERG, Tim. **Sistemas Distribuídos: Conceitos e Projeto. 5. ed.** Editora: Bookman, 2013.

GORDON. **Gordon-server.** <https://www.npmjs.com/package/gordon-server>. Acesso em 21 de novembro.

LANCE. **RealTime. Multiplayer. JavaScript**. <http://lance.gg/>. Acesso em 10 outubro 2017.

SEABRA, Rodrigo. **Uma ferramenta em realidade virtual para o desenvolvimento da habilidade de visualização espacial.** Escola Politécnica, Universidade de São Paulo, 2009.

COLPANI, Rogério, MURILLO, Homem R. P. **Realidade Aumentada e Gamificação: A tecnologia no auxílio de alunos com deficiência intelectual**. Novas Edições Acadêmicas, 2015.

ROSSI, E. G. **Ambiente de apoio ao desenvolvimento de aplicações distribuídas e reconfiguráveis utilizando agentes de busca e classificação inteligentes.** Biblioteca Digitais de Teses e Dissertações da USP, 2009.

MORAES, William Bruno. **Construindo aplicações com NodeJS**. Editora: Novatec, 2015.

TORRES, Zuñiga; CALOS, Juan. **Uma metodologia para o desenvolvimento de aplicações de realidade aumentada em telefones celulares utilizando dispositivos sensores**. Biblioteca Digitais de Teses e Dissertações da USP, 2008.