Aplicação da biblioteca OpenGL em um jogo desenvolvido no C# com integração do Arduino Uno e Ionic Framework.

Carlos Roberto dos Reis[1], João Paulo da Silva[1], Thiago Henrique da Silva[1], Jaqueline Corrêa Silva de Carvalho[2]

[1] Discentes do Curso de Ciência da Computação da Universidade José do Rosário Vellano (UNIFENAS) – Alfenas MG

[2] Docente do Curso de Ciência da Computação da Universidade José do Rosário Vellano (UNIFENAS – Alfenas MG); Mestre em Engenharia Elétrica – UNIFEI, Itajubá MG.

{carlosrobertodosreis11,thiago.guinhoo}@gmail.com, joaopauloexcel@hotmail.com, jaqueline.carvalho@unifenas.br

Abstract. *Being a specialist in a given programming language brings with it the best practical results through consolidated research, but the big question is when there is a need to explore the various features that a graphics library can offer for computing, as well as interact with it technologies in order to demonstrate the unification of different tool concepts around a single application, all for the benefit of the growth of computational exploration.*

New technologies have always been adopted by a broad market, bringing with them the challenges of innovations and integration, especially in the field of games, where there is an immense search by programmers for specific tools aimed at game development, which in turn can to enjoy the integration of the game itself with external components, be these logic circuits are sensors or leds, for example.

The purpose of this article is to demonstrate the integration of technologies, in a simple way, namely, the resources needed to build a shooting game that interacts externally with an embedded circuit (where it will send and receive data transmission through communication serial) as well as share the results obtained by the players in the game with a Mobile platform to analyze the score of the scores.

Resumo. *Ser especialista em uma determinada linguagem de programação trás consigo os melhores resultados práticos por meio de pesquisas consolidadas, mas a grande questão é quando há a necessidade de explorar os vários recursos que uma biblioteca gráfica pode oferecer para a computação, além de interagir com ela tecnologias de forma a demostrar a unificação de diferentes conceitos de ferramentas em torno de uma única aplicação, tudo isso, para benefício do crescimento da exploração computacional.*

Novas tecnologias sempre vêm sendo adotadas por um mercado amplo, trazendo com elas os desafios de inovações e integração, principalmente no ramo dos jogos, onde há uma busca imensa pelos programadores por ferramentas

específicas voltadas para o desenvolvimento de game, que por sua vez, podem usufruir da integração do jogo em si com componentes externos, sejam esses circuitos lógicos como sensores ou leds, por exemplo.

O intuito desse artigo é demonstrar a integração de tecnologias, de forma simples, a saber, de recursos necessários para construção de um jogo de tiro ao alvo que interaja externamente com um circuito embarcado (onde irá enviar e receber transmissão de dados por meio da comunicação serial), bem como compartilhar os resultados obtidos pelos jogadores no game com uma plataforma Mobile para análise do ranking dos pontuadores.

1. Introdução

Na atualidade, com os avanços tecnológicos do presente século, é comum pessoas de várias faixas-etárias optarem por passar determinado tempo de seus dias em conjunto com equipamentos eletrônicos, sejam eles smartphones, tabletes, notebooks, entre outros. Em vista disso, um dos meios que tem ganhado espaço no cotidiano dos indivíduos, principalmente para o público jovem, são os games. Alguns desses, por sua vez, apresentam inovações dinâmicas e reais do ambiente tecnológico que envolve os usuários.

O que muitos desses usuários desconhecem é que, por traz de interfaces cada vez mais parecidas com o mundo real, há um complexo algoritmo desenvolvido por profissionais da área da computação gráfica, onde muitos desses dedicam uma considerável parte de suas vidas na projeção e implantação de jogos.

Em meio a esse desenvolvimento, vários dos profissionais e empresas da área de *games* estão buscando meios externos para "automatizar" os jogos. Um exemplo é o Xbox 360, que possui um sensor de presença capaz de captar os movimentos dos jogadores posicionados a sua volta, cuja imagem é virtualizada dentro do ambiente gráfico do jogo, o que torna para o usuário um divertido e dinâmico meio de aproveitar o tempo livre com amigos e familiares.

Esse artigo tem como objetivo apresentar uma forma simples de aplicação de *game* utilizando a computação gráfica em conjunto com tecnologias externas capazes de interagir com o usuário de acordo com o decorrer do jogo.

Referente à aplicação, será desenvolvido um jogo de tiro ao alvo, cujo jogador, por meio do mouse do computador, movimentará a mira na tela, e, utilizando o teclado do dispositivo, conseguirá escolher o nível de dificuldade do jogo, inserir o nome do jogador, alternar o cenário da aplicação e atirar no alvo (que estará em constante movimento alternando sua velocidade para a esquerda e direita, de acordo com o nível do jogo preestabelecido pelo usuário).

2. Referencial teórico

2.1. Mercado promissor de games no Brasil

No Brasil, o mercado dos games é promissor, mas ainda é novo e, segundo (PERUCIA, BALESTRIN e VERSCHOORE, 2011) tem muito ainda a expandir no país no contexto global.

Embora a atuação dos games seja ainda nova no Brasil em relação ao exterior, há uma grande expectativa do aumento dessa área tecnológica em nosso território, mas para isso, segundo (LOBO, VERDI, ELIAS. 2012), é necessário o incentivo do governo em parcerias com empresas do ramo para o aumento do número de pessoas capacitadas a esse tipo de desenvolvimento.

2.2. Computação Gráfica

A computação gráfica utiliza um conjunto de métodos e técnicas para transformar figuras e imagens capturadas por equipamentos através da tecnologia em um dispositivo gráfico. Pode-se classificar aplicativos relacionados à computação gráfica como passivos, que apresentam dados em forma de figuras sem a interferência do usuário no processo e, também, como sendo interativos, utilizando dispositivos tecnológicos para apresentar e preparar imagens, permitindo ao usuário interagir de acordo com a sua necessidade. (SCALCO, 2005)

2.3. Biblioteca OpenGL

A biblioteca OpenGL (Open Graphics Library) é uma aplicação para dispositivos, idealizada para construção de interfaces. Atua de forma gráfica em modelagem e exibição tridimensional, sendo considerada veloz e portável para vastas plataformas. Sua utilização permite ao usuário manipular objetos gráficos de modo rápido, além de fornecer recursos de animação, impressão de textos na tela e perspectivas em 2D e 3D.

Essa biblioteca foi introduzida em 1992 pela Silicon Graphics para distribuir uma API (Interface de Programação de Aplicação) gráfica. Dessa forma, serviria como mediador entre o processo de modelagem geométrica de objetos, situados em um nível de abstração mais elevado, e as rotinas de exibição e de processamento de imagens.

Existem os mesmos nomes e parâmetros em todos os sistemas operacionais nos quais a OpenGL foi implementada, e, por sua vez, produz o mesmo efeito de exibição em cada um desses sistemas. (BRITO. 2015)

2.4. Primitivas Geométricas

As primitivas geométricas na OpenGL são constituídas a partir de seu vértices, onde um vértice é representado em coordenadas homogêneas (x, y, z, w). Os cálculos internos são realizados com pontos definidos no espaço tridimensional, ou seja, se houver pontos bidimensionais especificados pelo usuário, são trabalhados como pontos tridimensionais, onde a coordenada z é igual à zero.

Para ser localizado um ponto P(x,y,z) na tela, onde x = 0.2, y = -0.1 e z = -0.8, por exemplo, primeiramente, é necessário encontrar a interseção entre os eixos „x" e „z", somente depois, projeta-la até o ponto y.

Para isso, traça-se uma reta „z" paralelamente ao próprio eixo „z", sendo essa sobreposta ao ponto „x". Posteriormente, projeta-se uma reta „x" em paralelo ao eixo „x" sobreposto ao ponto „z". A partir do ponto de interseção referente às retas „x" e „z", traça-se outra reta „y" paralelamente ao eixo „y", indo à altura do ponto da coordenada „y" especificada. Na Figura 1, percebe-se como a projeção ocorre sobre os pontos referenciais 3D:

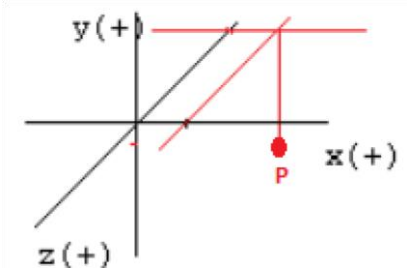

Figura 1. Projeção de coordenadas 3D

Existem alguns cuidados quanto à definição de polígono na OpenGL, assim, é necessário considerar que um polígono deverá ser sempre convexo e simples (não poderá haver interseção das suas arestas). A função glVertex() especifica um vértice do polígono.

Com o OpenGL, é possível definir polígonos não simples, como côncavos ou com furos. A derivação de qualquer polígono pode ser formado a parti de união de polígonos convexos com algumas rotinas mais complexas derivadas das primitivas básicas que são fornecidas na GLU (OpenGL Utility Library), onde essa biblioteca utiliza funções que estão disponíveis em todas as implementações da OpenGl. (BICHO,2002)

A Figura 2 demonstra os argumentos que poderão ser inseridos no glBegin() e a ordem que serão associados os vértices.

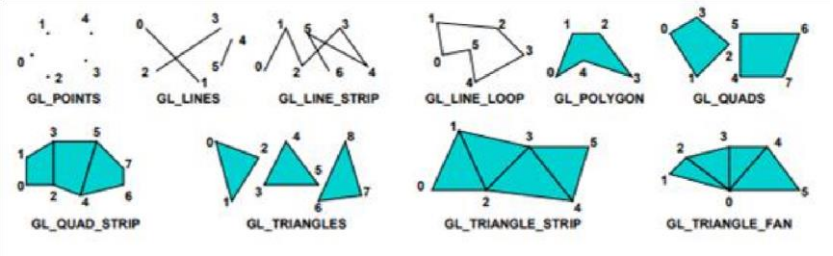

Figura 2. Primitivas gráficas da Open GL

2.5. Iluminação

Para promover um realismo a objetos na cena, a OpenGL leva em consideração alguns aspectos como, tipo de fonte de iluminação que está sendo utilizada na cena e as propriedades do material para cada superfície. Efeitos como reflexão de luz e sombra são complexos, assim, para programar o modelo de iluminação na OpenGL, decompõem-se o raio luminoso nos componentes primárias RGB, assim a cor para uma fonte luminosa é relativo a quantidade de intensidade total do componente emitido.

```
// Ativa o uso da luz ambiente
Gl.glLightModelfv(Gl.GL_LIGHT_MODEL_AMBIENT, luzAmbiente);

// Define os parâmetros da luz de número 0
Gl.glLightfv(Gl.GL_LIGHT0,
Gl.GL_AMBIENT, luzAmbiente);
Gl.glLightfv(Gl.GL_LIGHT0, Gl.GL_DIFFUSE, luzDifusa);
Gl.glLightfv(Gl.GL_LIGHT0, Gl.GL_SPECULAR, luzEspecular);
Gl.glLightfv(Gl.GL_LIGHT0, Gl.GL_POSITION, posicaoLuz);
```

Figura 3. Codificação do algoritmo de iluminação

No código da Figura 3, exibida anteriormente, é localizado as variáveis “luzAmbiente”, “luzDifusa” e “LuzEspetacular”, onde ficam responsáveis pela definição dos parâmetros de divisão da luz, pelo fato que o OpenGl considera que ela é dividida nestes três componentes independentes.

•**Ambientes**: Resultado da luz refletida no ambiente, é uma clareamento que vem de todas as direções;

•**Difusa**: Luz que vem de uma direção e atinge a superfície e é refletida em todas as direções; assim, parece possuir o mesmo brilho independentemente de onde a câmera esteja posicionada;

•**Espetacular**: Luz que vem de uma direção e tende a ser refletida em uma única direção;

Na OpenGL, os modelos de iluminação na cena podem vir de várias fontes, mas sendo controladas de forma individual, podendo ser provenientes de uma determinada direção ou posição, enquanto outras podem estar dispersas na cena. (MANSSOUR, [2010])

2.6. Linguagem C#

No final dos anos 90, a Microsoft possuía diversas linguagens para resolver diversos tipos de problemas, porém, para cada vez que um desenvolvedor mudava de linguagem para solucionar tal problema, era necessária, além do conhecimento das especificações da nova linguagem, a adaptação das APIs e Bibliotecas. Buscando minimizar esse problema de adaptação, foi desenvolvido o .NET, uma plataforma da Microsotf onde se fez base para todas as linguagens fornecidas pela empresa.

Assim, para algum profissional mudar de linguagem, apenas necessitou saber das especificações da mesma, padronizando a estrutura de todas as bibliotecas e APIs. Com isso, a linguagem C# foi lançada em 2002, em um projeto que visava desenvolver uma tecnologia leve, de fácil aprendizado e orientada a objetos. (WAGNER, 2015)

2.7. Arduino

O Arduino, nada mais é, do que uma placa de tamanho reduzida e de fácil manipulação para criação de projetos tecnológicos embarcados. Com isso, há uma vasta comunidade de desenvolvedores que o utilizam em vários projetos para diversas finalidades.

Essa placa, possui hardware e software open-source, onde há inúmeros sensores e componentes que podem ser utilizados junto a ela, facilitando a criação de projetos e alterações no decorrer do desenvolvimento. Possui, também, IDE gratuita e linguagem baseada C/C++. (MCROBETS, 2018)

2.8. Ionic

Lançado em 2013, trata-se de um framework híbrido, isto é, você pode desenvolver um aplicativo para várias plataformas através de um único código fonte. De forma simplificada, podemos definir um aplicativo híbrido como um aplicativo nativo que exibe páginas web embutidas. A codificação dos aplicativos híbridos se faz por meio de HTML, CSS e JavaScript, e, através de ferramentas, esses códigos são convertidos para plataformas como Android, Windows Phone e iOS. Consequentemente, qualquer indivíduo com habilidades em desenvolvimento web poderia criar um aplicativo utilizando-se de tecnologias híbridas como esta.
Além de HTML, CSS e JavaScript, o Ionic ainda faz uso do Angular Framework, da linguagem TypeScript e da ferramenta Cordova, sendo que tais recursos serão apurados logo adiante.

Para que o software final seja exibido nos aspectos visuais do sistema operacional alvo, é feita uma "tradução" de códigos através de uma série de plug-ins disponibilizados por ferramentas como o Cordova e o próprio Ionic Framework. (BRADLEY, 2013).

2.9. Cordova

Ferramenta de código aberto para a construção de aplicações nativas a partir de códigos HTML, CSS e JavaScript.

O triunfo do Cordova se dá pelo fato de que ele disponibiliza um conjunto de APIs que fornecem acesso ao hardware e recursos nativos de um sistema, tais como câmera, acelerômetro, lista de contatos e qualquer outro recurso da camada de hardware ou software nativo.

Para fazer tudo isso, o Cordova utiliza um recurso do sistema chamado WebView, cujo funcionamento se assemelha ao de um navegador de internet, porém, sem menus, botões e barra de endereços. Assim, nossa aplicação final tem aparência muito semelhante à de um aplicativo nativo. (LANGE JUNIOR; MERCADO, 2018)

2.10. Angular

O Angular é um Framework front-end criado pelo Google que utiliza um conjunto de códigos HTML, CSS e TypeScript, sendo esse ao final compilado para JavaScript.

O desenvolvimento em Angular se faz por meio de componentes, os quais combinam partes de código HTML chamados templates, e classes dotadas da anotação "@Component", essas ficam a cargo do gerenciamento do conteúdo na página Web.

A união destes dois conceitos citados anteriormente estabelece um componente Angular. (GUEDES, 2017)

2.11. NodeJS

O NodeJS é uma plataforma para a linguagem de programação JavaScrip, que utiliza o Engine JavaScript do navegador Google Chrome (motor V8 open-source e escrito em C++), projetado para criar, de forma fácil e rápida, aplicativos de rede escalonáveis com o tempo de execução orientado a eventos assíncronos, podendo tratar muitas conexões simultaneamente.

A plataforma utiliza uma arquitetura não-bloqueante, que apresenta um bom desempenho com o consumo de memória, utilizando de forma eficiente os recursos dos servidores, facilitando, assim, a execução paralela e o aproveitamento de recursos de hardware e software. (MORAES, 2018).

2.12. Firebase

O Firebase é uma plataforma do Google que pode ser utilizada para vários fins, principalmente como banco de dados online não relacional (NO-SQL). Esse tipo de banco de armazenamento é em nuvem e não tem a utilização de chaves primárias, ligação direta entre tabelas, entre outros, conforme existe no banco de dados tradicional.
Esse mecanismo, também, auxilia a expansão de empresas de programação por desenvolver rapidamente aplicativos de alta qualidade. Ele fornece uma plataforma de ferramentas e serviços, que com apenas alguns passos, obtém-se funcionalidades, como o Push Notification configurado e em produção. (ADRIANO, 2018)

3. Metodologia

Essa aplicação foi elaborada com a missão de desenvolver um jogo de tiro ao alvo em conjunto com sistemas embarcados, aumentando assim a interação do jogador com o meio externo, além de dar possibilidade de consulta das pontuações obtidas por ele através de uma aplicação Mobile.

3.1. Estrutura da aplicação

A estrutura geral da aplicação foi constituída em cinco etapas expostas na Figura 4 e detalhadas posteriormente.

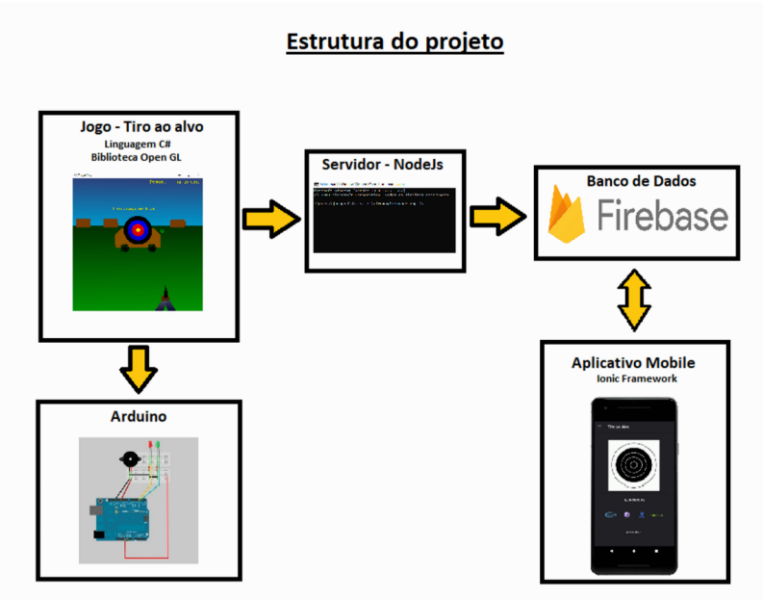

Figura 4. Esboço do projeto

1ª etapa: Aplicação do game, onde é executado o jogo do tiro ao alvo diretamente no desktop, conforme demonstra a Figura 5.

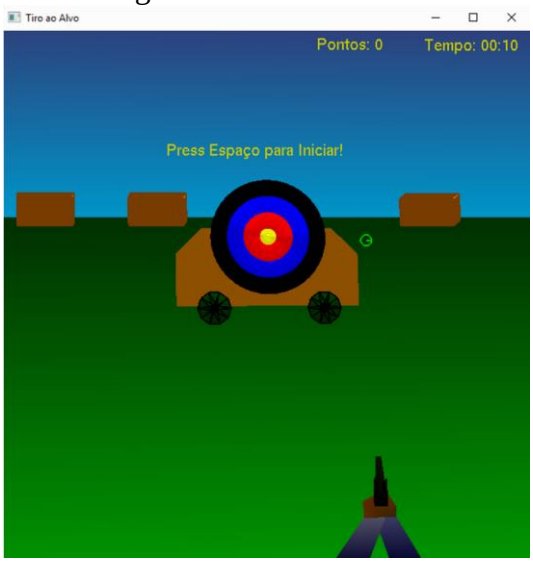

Figura 5. Esboço do projeto

2ª etapa: Aplicação embarcada no Arduino para acionamento dos leds e buzzer por meio dos tiros efetuados no jogo para demonstrar aos jogadores, por meio externo, o acerto e erro do alvo, o que tornou a aplicação mais dinâmica, conforme se esperava.

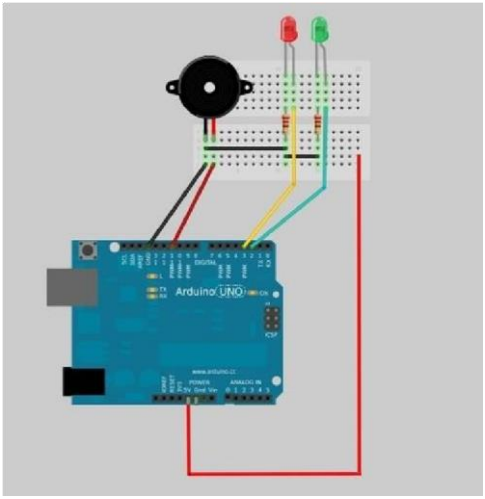

Figura 6. Protótipo do circuito utilizado no Arduino

3ª etapa: Aplicação do Servidor. No final de cada jogo, a aplicação em C# salva o nome do jogador e sua respectiva pontuação em um arquivo externo, na própria pasta do game. Esse arquivo é acessado pela aplicação do NodeJs, que faz a leitura e o envio dessas informações para o Firebase, gerando, assim, um novo registro no armazenamento em nuvem (que posteriormente será acessado pela aplicação Mobile).

4º etapa: Foi criado um projeto no Firebase para servir como banco de dados para a aplicação como um todo, para onde o NodeJs enviará os dados obtidos dos competidores, o que o torna a fonte de armazenamento de consultas e modificações dos dados realizados pelo aplicativo Mobile.

5ª etapa: Aplicação Mobile desenvolvida em Ionic, que consulta as informações obtidas pelos jogadores hospedadas no Firebase e as apresentam para o próprio usuário na tela de classificação do aplicativo.

3.2. Funcionamento da lógica do jogo

O alvo se movimenta por meio de uma variável „alvox‟, (representa a posição „x‟ dele na tela). Essa é incrementada, atingindo um determinado limite, passa a ser decrementada, ocasionando o movimento do alvo na tela para a esquerda e direita.

Referente à mira, para ela se movimentar, utiliza-se as coordenadas reais do mouse no monitor, o que torna impossível visualiza-la na tela do console no primeiro momento.

Para resolver o problema exposto anteriormente, foi necessário manipular as coordenadas multiplicando o valor da coordenada „x‟ por (0.0010) e, ainda, subtrair -0.3 (para o cursor invadir a parte negativa de „x‟ no console (do cento para a esquerda)).

Para a coordenada „y‟ do mouse, necessitou-se dividi-la por (-419) para encontrar o seu valor entre (0.0) e (-0.6) (limite da parte superior e inferior da janela do jogo no eixo „y‟, respectivamente) para, assim, a mira ser exibida na tela.

Também, foi criada uma equação para o acerto do tiro no alvo. Trata-se da raiz quadrada da soma de duas potencias elevadas ao quadrado, uma subtraindo o valor do eixo „x‟ da variável „alvox‟ por 0.5 (valor que a variável é inicializada para fixar previamente o alvo no centro da tela) subtraindo a isso o valor atual da variável „mouseX‟ (representando a coordenada do eixo „x‟ da mira).

Já a outra potência, subtraiu-se -0.27 (que é o valor da variável „mouseY‟ quando essa está no centro da tela do console) e, logo em seguida, subtraiu-se novamente a própria variável “mouseY”(onde realiza a verificação da mira, se está dentro ou fora do eixo do alvo na posição „y‟).

Resultado: Raiz(((AlvoX - 0.5 – mouseX) ^ 2) + ((- 0.27 – mouseY) ^ 2)).

No ato de depurar o código, notou-se que o resultado dessa equação sempre resulta menor que (0.01) quando o alvo é atingido na parte amarela (centro), menor que (0.045) se atingido na parte vermelha, menor que (0.074) se atingido na parte azul e, menor que 0.1 se atingido na parte preta. Acima desses valores, logo, concluiu-se que o tiro não acertou a área do alvo. A Figura 7 exibe a codificação que calcula o tiro no centro do alvo.

```
//Cálculo to tiro na cor amarela (centro) do alvo
if (Math.Sqrt((Math.Pow(((alvox - 0.5)-mouseX), 2) + Math.Pow((-0.27 - mouseY), 2))) < 0.01)
{
    Enviar("A");
    bitAcerto = true;
    pontos=pontos+10;
    pontuacao = "+10";
    Thread t = new Thread(BitAcerto);
    t.Start();
}
```

Figura 7. Codificação do cálculo ao acertar o centro alvo

A mesma regra é aplicada para as outras cores do alvo, alterando-se apenas o valor de comparação localizado após o operador “<”, dentro do comando condicional “if”.

Na Figura 8 é apresentada a tela inicial do aplicativo Mobile, na qual os competidores realizarão as consultas de suas respectivas pontuações na classificação dos pontuadores.

Figura 8. Aplicativo do Tiro ao alvo

4. Resultados e Discussões

Perguntas como: “A integração de todas as tecnologias propostas por esse projeto atingiram seus objetivos? As aplicações em si corresponderam ao que era esperado no findar dessa pesquisa em conjunto com a biblioteca OpenGL?” podem ser respondidas com a afirmação de êxito.

Através de um de uma aplicação em C#, foi desenvolvido um jogo de tira ao alvo. Nesta aplicação, foi utilizado a biblioteca OpenGL, para que fosse possível desenvolver todos os elementos dos cenário do jogo.

Na pasta onde se localiza o executável do jogo do tiro ao alvo, ficou também o arquivo servidor, desenvolvido em NodeJs, responsável pelo envio dos resultados do jogador para o Firebase, sendo assim, os resultados do jogador somente serão enviados para o ambiente Web caso esse servidor esteja acionado na máquina do usuário, entretanto, o jogo funcionará normalmente de forma independente, porém, sem ele, não ocorre o envio dos resultados obtidos pelo usuário para a aplicação Mobile.

Ao executar o jogo, o usuário se depara com a tela de coleta do nome e do nível que ele deseja estar jogando. É obrigatória a inclusão dessas informações para que o botão de inicialização do jogo seja acionado. Vale ressaltar, também, que o nível do jogo escolhido nesse primeiro momento é o que define a velocidade de movimentação do alvo na tela.

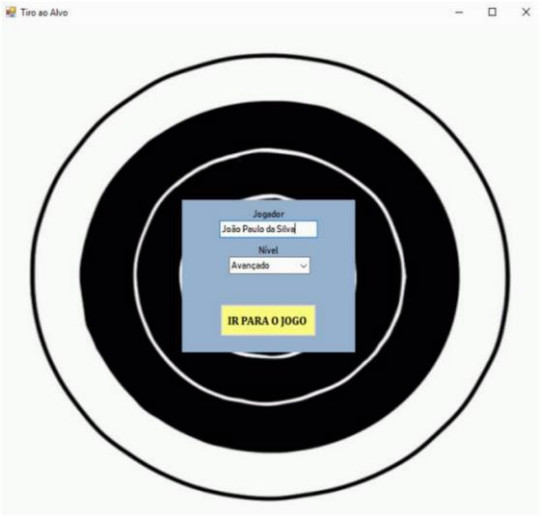

Figura 9. Formulário inicial do jogo

Após iniciar o jogo, de fato, é apresentado a tela contendo todo o algoritmo gráfico fornecido pela biblioteca OpenGL em conjunto com a linguagem C#.

Referente à movimentação do alvo, a cada rodada do jogo, ele estará por 10 segundos movendo-se para a direita e esquerda, tendo que ser acertado pela mira do atirador o maior número de vezes possível dentro desse tempo determinado, o que ocasiona o acúmulo de pontos para o jogador no final dessa execução.

Durante esse tempo de 10 segundos de cada jogo, a aplicação C# enviará as respostas de acerto e erro advindas do jogador no momento dos tiros para uma porta serial da máquina do usuário, o que acionará o buzzer (emitindo um sinal sonoro agudo para acertos e graves para erros), além dos leds verde e vermelho, respectivamente também para os acertos e erros.

É importante destacar que, se o jogador acertar a cor amarela do alvo (centro), marcará +10 pontos, a parte vermelha +6 pontos, a parte azul +3 pontos, a parte preta +1 ponto e errando o alvo perderá -5 pontos. A seguir está a Figura 10 que demonstra dois momentos, acertando e errando o tiro no alvo.

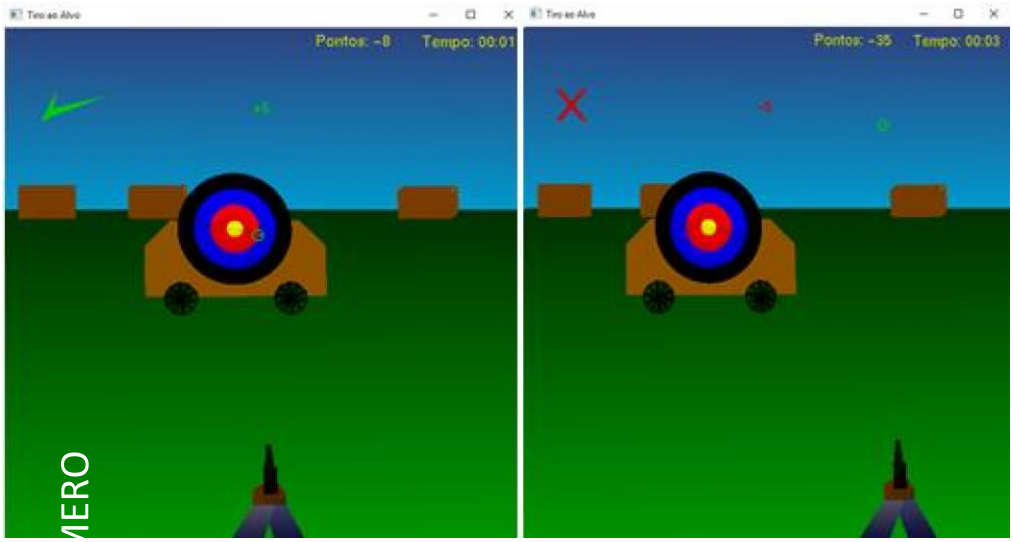

Figura 10. Momentos dos tiros no alvo

Em adição, após o término do tempo do jogo, a aplicação do servidor NodeJs envia o nome do jogador e a pontuação que ele fez para o Firebase, ou seja, o servidor funciona como ponte de troca de informações ente o jogo e o aplicativo Mobile.

Por fim, no aplicativo Mobile, o jogador consegue ter acesso ao ranking dos jogadores ordenado de forma decrescente, de acordo com o número de pontos atingidos no momento do jogo, conforme ilustra a Figura 12.

Figura 11. Classificação no aplicativo Mobile

Pode-se, também, excluir ou editar algum dos jogadores, inserir foto de perfil, realizar consultas específicas de jogadores ou, até mesmo, efetuar consultas do manual do jogo.

Figura 12. Tela de ajuda do aplicativo Mobile

5. Conclusão

Em suma, essa pesquisa atingiu seu objetivo de integrar todas as tecnologias propostas com sucesso em conjunto com uma eficiente exploração dos recursos trazidos pela biblioteca gráfica OpenGL.

Em primeiro lugar, o desenvolvimento utilizando essa biblioteca permitiu a criação de todo cenário na perspectiva 3D, deixando assim, muito mais interativa e realista a aplicação do jogo.

Em segundo lugar, foi realizada a integração do sistema embarcado, utilizando a placa Arduino, onde ofereceu uma considerável dinâmica externa aplicável por meio da interação dos componentes de leds e buzzer com o jogo.

Em terceiro lugar, na listagem do ranking utilizando a aplicação em Ionic, foi fornecida ao jogador a possibilidade de analisar a própria atuação no jogo perante aos outros competidores.

Em continuidade, vale destacar a importância do servidor NodeJs, que funcionou perfeitamente no âmbito de intermedir os resultados obtidos localmente com o aplicativo Mobile.

Ficou evidente, por tanto, que por mais que as tecnologias sejam diferentes em conceitos e formas de interação, o grande desafio idealizado e atingido foi à integração delas em torno de uma só finalidade, que foi a de proporcionar ao usuário final um jogo além da tela do console, onde, além da aplicação da OpenGL da computação gráfica, foi introduzido sistemas embarcados, Servidor NodeJS para o back-End e Aplicativo Mobile.

6. Referências

PERUCIA, Alexandre; BALESTRIN, Alsones & VERSCHOORE. Coordenação das atividades produtivas na indústria brasileira de jogos eletrônicos: hierarquia, mercado ou aliança? Produção, v.21, n. 1, p-64-75, jan./mar. 2011

LOBO, Clever Zuin; VERDI, Luis Cyrilo Ganassim; ELIAS, Paulo César Elias. Um estudo exploratório sobre o mercado de jogos eletrônicos no Brasil. Revista Conteúdo; Capivari-SP, v.2, n.1, p.79, jan./jul. 2012 – ISSN 1807-9539

SCALCO, Roberto. Introdução à Computação Gráfica. 1 ed. [s.l.]: Roberto Scalco, 2005. Disponível em: < https://books.google.com.br/books?id=zSOdvdYd9BUC>. Acesso em Junho de 2019.

BRITO, Agostinho. Introdução à computação gráfica com OpenGL. Disponível em: <https://agostinhobritojr.github.io/tutorial/opengl/>. Acesso em: 24 jun. 2019.

BICHO, Alessandro L et al. Programação Gráfica 3D com OpenGL, Open Inventor e Java 3D. Revista Eletrônica de Iniciação Cientifica (REIC). [s.l.], v.2, n.1, p. 2-3, Mar/Mar. 2002.

MANSSOUR, Isabel Harb. Introdução à OpenGL: Utilizando Luzes. [2010]. Disponível em: <https://www.inf.pucrs.br/~manssour/OpenGL/Iluminacao.html>. Acesso em: 20 jun. 2019

WAGNER, Bill. Introdução à linguagem C# e ao .NET Framework. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/csharp/getting-started/introduction-to-thecsharp-language-and-the-net-framework>. Acesso em: 24 jun. 2019.

WARREN, Genevieve; Bem-vindo ao IDE do Visual Studio; Site Oficial da Microsoft. Março/2019; Disponível em: <https://docs.microsoft.com/pt-br/visualstudio/getstarted/visual-studio-ide?view=vs-2019>. Acessado em Abril de 2019.

MCROBETS, Michael. Arduino Básico: Tudo sobre o popular microcontrolador Arduino. 2 ed. [s.l.]: Novatec, 2015.

BRADLEY, Adam. Where does the Ionic Framework fit in? 2013. Disponível em: <https://ionicframework.com/blog/where-does-the-ionic-framework-fit-in/>. Acessado em Junho de 2019.

LANGE, Junior, NERI, Norberto; MERCADO, Neyza Bibiana Guzmán. Vantagens e desvantagens para utilização do Ionic para dispositivos móveis. Revista Científica Semana Acadêmica. Fortaleza, v.1, n.145. Disponível

em:<https://semanaacademica.org.br/artigo/vantagens-e-desvantagens-da-utilizacaodo-ionic-framework-para-o-desenvolvimento-de>; Acessado em Maio de 2019.

GUEDES, Thiago. Crie aplicações com Angular: O novo framework do Google. 1 ed. São Paulo, SP: Casa do Código, 2017. Disponível em: <https://www.casadocodigo.com.br/products/livro-angular>; Acessado em Abril de 2019.

MORAES, William Bruno. Construindo aplicações com NodeJS. 2 ed. [s.l.]: Novatec Editora, 2018. Disponível em: <https://books.google.com.br/books?id=zBdiDwAAQBAJ&printsec=frontcover&hl=pt-BR&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false>; Acessado em Abril de 2019.

BANZI, Massimo; SHILOH, Michael. Primeiros Passos com o Arduino: A plataforma de prototipagem eletrônica open sorce. 2 ed. [s.l.]: Novatec, 2015.

ADRIANO, Thiago S.. Introdução ao Firebase. Disponível em: <https://medium.com/@programadriano/introdu%C3%A7%C3%A3o-ao-firebasebd59bfd03f29>. Acesso em: 24 jun. 2019.