Revista brasileira de iniciação científica
ISSN: 2359-232X

DESENVOLVIMENTO DE JOGO 3D DE FUTEBOL DE ROBÔS

DEVELOPMENT OF ROBOT SOCCER 3D GAME

DESARROLLO DE JUEGO 3D DE FÚTBOL DE ROBOTS

Diego Eduardo Carvalho[1]
Antonio Valerio Netto[2]

Resumo: O RoboGol é um sistema criado para a realização de partidas de futebol de robôs. Inspirado no futebol de mesa, popularmente conhecido como "totó" ou "pebolim", o sistema eletrônico conta com uma mesa energizada que alimenta de um a quatro robôs. Estes robôs são controlados individualmente por meio de controles, modelo Arcade, fixados na mesa. O desenvolvimento para o formato online (*web*) foi à alternativa encontrada para atingir um maior número de usuários. Este artigo descreve as tecnologias utilizadas na implementação, à solução gerada e as dificuldades encontradas no processo de desenvolvimento.
Palavras-chave: Game 3D. futebol de robôs. Unity. RoboGol.

Abstract: RoboGol is a system designed for robot football matches. Inspired by table soccer, popularly known as "totó" or "pebolim", the electronic system has an energized table that feeds from one to four robots. These robots are controlled individually by means of controls, arcade model, fixed to the table. The development for the online format (*web*) was the alternative found to reach a greater number of users. This article describes the technologies used in the implementation, the solution generated and the difficulties encountered in the development process.
Keywords: Game 3D. Robots soccer. Unity. RoboGol.

Resumen: RoboGol es un sistema creado para la realización de partidos de fútbol de robots. Inspirado en el fútbol de mesa, popularmente conocido como "totó" o "pebolín", el sistema electrónico cuenta con una mesa energizada que alimenta de uno a cuatro robots. Estos robots son controlados individualmente por medio de controles, modelo Arcade, fijados en la mesa. El desarrollo para el formato online (*web*) fue la alternativa encontrada para alcanzar un mayor número de usuarios. Este artículo describe las tecnologías utilizadas en la implementación, la solución generada y las dificultades encontradas en el proceso de desarrollo.
Palabras-clave: Game 3D. fútbol de robots. Unity. RoboGol.

Envio 11/09/2017 **Revisão 14/09/2017** **Aceite 05/10/2017**

[1] Graduado. ICMC/USP. E-mail: contato@xbot.com.br.
[2] Doutor. ICMC/USP. antonio.valerio@pq.cnpq.br

Revista brasileira de iniciação científica
ISSN: 2359-232X

Introdução

O RoboGol (Robogol, 2015) foi um sistema criado para a realização de partidas de futebol entre robôs móveis. O equipamento é constituído de uma mesa eletrônica no formato de campo de futebol com dimensões 2,85m x 1,60m x 0,90m pesando aproximadamente 160 kg, com um a quatro robôs pesando em torno de 4 kg cada um, controlados por joysticks modelo Arcade fixados na mesa, além de uma bola de golfe. O projeto teve como objetivo realizar o desenvolvimento do RoboGol para o formato *online* (*web*). Por meio dessa iniciativa, o sistema irá simular em 3D, o ambiente real do jogo tornando possível que os usuários possam disputar uma partida mesmo distante da versão física.

Para a execução desse projeto optou-se pela utilização da ferramenta Unity 3D versão 3.0 como motor de jogos (Unity 3D, 2015). O motor de jogos, mais conhecido pelo termo em inglês "*game engine*", é um programa de computador ou um conjunto de bibliotecas que fornece uma abstração dos detalhes da construção de um jogo digital. As funcionalidades tipicamente encontradas em uma *engine* são: renderização gráfica 2D e 3D, simulação dos aspectos físicos e detecção de colisão. Fornece também uma abstração de hardware permitindo ao desenvolvedor criar jogos sem se preocupar com a arquitetura da plataforma fim (Goldstone, 2009; Blackman, 2011; Wittayabundit, 2011).

A escolha da Unity 3D foi baseada no fato de possuir características que auxiliam o programador a concluir seu trabalho de forma rápida e fácil, além de ter uma versão gratuita para desenvolvedores iniciantes (Unity 3Da, 2015). A ferramenta não é apenas uma *engine*, contando também com um ambiente integrado de desenvolvimento e com um sistema de publicação que se enquadra na categoria de *midleware*. Middleware é o termo criado para designar camadas de software que não constituem diretamente aplicações, mas que facilitam a criação das mesmas (Watson, 2002; Greene, 2002). Dentre as características encontradas na ferramenta é possível citar que os jogos digitais podem ser criados utilizando as linguagens de programação: Boo, JavaScript ou C#. É compatível com diversos navegadores *web* e programas de modelagem tridimensional. Além disso, possui uma interface gráfica, sistema de colisão, iluminação e som. Outro ponto forte da Unity 3D é o fato de ser multiplataforma, podendo ter jogos desenvolvidos tanto para *desktops*, *mobile* ou *web*.

Revista brasileira de iniciação científica
ISSN: 2359-232X

Para a codificação do jogo 3D, a linguagem de programação escolhida foi o C#. Trata-se de uma linguagem orientada a objetos desenvolvida pela Microsoft como parte da plataforma ".NET". Apesar da UNITY já possuir ambiente para criação e edição de código integrado ao seu ambiente, foi utilizado o ambiente de desenvolvimento do Microsoft Visual Studio 2010® (MICROSOFT, 2015) por possuir um maior material de consulta na Internet. O Visual Studio 2010® também foi utilizado como ferramenta para a criação dos diagramas UML (*Unified Modeling Language*) que serviram de base para o desenvolvimento. O banco de dados utilizado foi o PostgreSQL versão 8.4. O PostgreSQL é um SGBD (Sistema de Gerenciamento de Banco de Dados) objeto-relacional *open source*, que possui funcionalidades sofisticadas como controle de concorrência, *tables spaces*, cópias de segurança, tolerância a falhas, entre outras.

Com as ferramentas definidas fez-se a análise de requisitos seguindo o padrão da UML e modelando o jogo com a aplicação de técnicas de análise orientada a objetos. A UML permite ao engenheiro de software traduzir as regras de negócios por meio de métodos unificados com modelos de análise usando regras definidas por Grady Booch, James Rumbaugh e Ivar Jacobson (Presman, 2002). Os métodos de análise orientados a objetos definem as regras do problema como um conjunto de objetos que possuem relação ou não entre si, podendo ou não herdar atributos uns dos outros e que se comunicam por meio de mensagens (Presman, 2002).

Neste artigo são apresentadas as atividades desenvolvidas, assim como os resultados obtidos e as dificuldades encontradas durante a execução do projeto. Também são descritas as considerações finais sobre o trabalho realizado, além de comentar sugestões de trabalhos futuros. Por fim, são apresentadas as referências citadas ao longo do texto.

Desenvolvimento do trabalho

O projeto teve como objetivo projetar e desenvolver o jogo RoboGol para a plataforma *web*, ou seja, a solução final deveria ser executada por meio de um navegador *web*. O jogo digital deveria se assemelhar ao correspondente real, que é utilizado para fins de entretenimento. Atentou-se tanto para parte visual, quanto para a ambientação do jogador, no que diz respeito à jogabilidade, visto que um dos objetivos do jogo é incentivar e estimular os jogadores a realizar a recorrência de jogar novamente. A meta de desenvolvimento também incluía que todas as informações das partidas e dos jogadores fossem armazenadas no banco de

Revista brasileira de iniciação científica
ISSN: 2359-232X

dados e os desempenhos e estatísticas de cada usuário, exibidos na área de ranking e estatística de partidas do site do jogo.

Para tanto, o jogo 3D deveria permitir que o jogador controlasse o robô virtual com movimentos para frente, para trás e de rotação no próprio eixo. Seriam quatro robôs virtuais idênticos no formato, mas dividido por cores em dois times. Deveria estar presente também o campo de futebol com dois gols e uma bola branca. Além de permitir a exibição de um marcador de tempo da partida e de um marcador de pontuação dos gols separado por time. Outros detalhes não funcionais contemplados foram: uma boa iluminação ambiente, a visão completa de toda a mesa e dos robôs nela posicionados, nitidez na diferenciação de cores entre os times, menu com informações sobre os controles necessários para a realização das partidas e a visualização do desempenho individual de cada jogador após cada partida (ranking).

Para o desenvolvimento do jogo RoboGol virtual foi realizado um estudo sobre as funcionalidades e as características da *engine Unity* 3D. Com uma base de conhecimento adquirida, deu-se início ao levantamento de requisitos do projeto juntamente com uma modelagem de software por meio de diagramas de estado e de classe baseado nos padrões UML. Após esta etapa concluída foram criados os documentos de definição de layout e itens que estariam presentes em cada tela apresentada durante a execução do jogo. Com toda a documentação concluída deu-se início a codificação e testes do jogo 3D.

Unity 3D

Como a Unity 3D possui um ambiente integrado de desenvolvimento, o ponto de partida iniciou-se com a ambientação das telas e estrutura para a criação de jogos (Figura 1). Os jogos são montados por meio de cenas que são como blocos ou etapas do jogo. Realizando o desenvolvimento por cenas é possível economizar processamento, pois somente os elementos presentes na cena ativa consumirão o processador.

Revista brasileira de iniciação científica
ISSN: 2359-232X

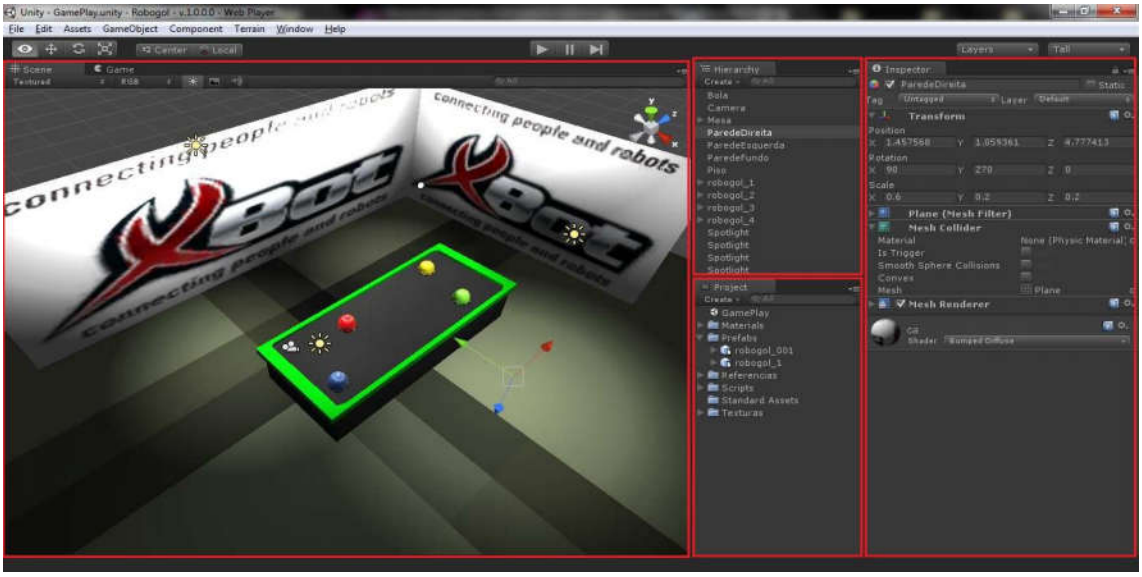

Figura 1 - Ambiente de Desenvolvimento do Unity 3D.

A IDE (*Integrated Development Environment*) possui uma interface simples e amigável que tem por objetivo facilitar o desenvolvimento dos jogos. Ela é composta por quatro telas, cada uma com uma função especifica que são exibidas por padrão na inicialização do sistema. São elas *Scene, Hierarchy, Project, Inspector*. Estas janelas possuem as principais informações para o desenvolvedor realizar seu trabalho.

A janela *Scene* é a principal no ambiente de desenvolvimento. Ela é a janela que exibe todos os elementos da aplicação, possibilitando a manipulação dos mesmos por meio de rotações, posicionamento e alteração de sua escala. Nessa tela também é possível navegar por entre os objetos permitido que se tenha uma visão de diversos ângulos e distâncias.

A janela *Hierarchy* exibe todos os elementos presentes em uma determinada cena em edição. Esses elementos são organizados em uma hierarquia e exibidos em forma de árvore de visualização que demonstra a composição da cena.

A janela *Project* é responsável pela manipulação dos arquivos que representam um projeto, que contém os códigos (*scripts*), texturas, modelos tridimensionais, arquivos de áudio e um componente próprio da *engine*, os *prefabs*. *Prefabs* são objetos armazenados que podem ser utilizados inúmeras vezes em qualquer cena do projeto. A partir do objeto original é possível instanciar qualquer quantidade de objetos que irão apresentar todas as características e

Revista brasileira de iniciação científica
ISSN: 2359-232X

comportamentos do objeto base. Esse comportamento é similar a criar uma classe e instanciar vários objetos em uma linguagem de programação orientada a objetos. A janela Project é organizada de forma idêntica ao sistema de arquivos do computador, permitindo até que algumas alterações ocorram diretamente nos diretórios do sistema operacional refletindo diretamente no projeto.

A janela *Inspector* apresenta os componentes de um determinado objeto. A *Unity* 3D constrói seus elementos por meio de composição. Dessa forma, ao selecionar algum item em uma cena são exibidos todos os componentes desse objeto, como por exemplo, componentes de *Transform*ação, de renderização e de colisão. Cada componente é constituído de atributos que são valores editáveis que influenciam diretamente o comportamento ou característica do objeto.

A partir da familiarização desse ambiente de desenvolvimento o próximo passo foi identificar os componentes que determinam um jogo desenvolvido nessa ferramenta. O primeiro elemento e o principal é o *Game Object*, esse componente é um contêiner genérico que serve de base para praticamente todos os outros componentes de um jogo. O *Game Object* funciona como um repositório de funcionalidades. Cada componente acrescenta uma funcionalidade ou comportamento ao objeto. Esse componente pode ser um objeto de iluminação, uma geometria de colisão e até uma textura. Esse objeto pode se tornar qualquer coisa na cena, sendo possível criar uma câmera simples até uma espaçonave interestelar dependendo de quais elementos o compõe. Todo o *Game Object* possui como característica inicial o componente *Transform*, responsável pelo posicionamento, orientação e escala no sistema referencial da cena. É a partir *Game Object* que são criados os *prefabs*.

Apesar do editor da UNITY possuir ferramentas para a criação de objetos básicos como esferas, cubos e cilindros, e ter ferramentas para coloração, efeitos de som e criação de partículas. Para os jogos mais sofisticados é essencial à criação de objetos complexos como, por exemplo, humanoides, veículos, ou sons das mais diversas formas. A UNITY permite que elementos possam ser criados em programas de modelagem 3D especializados e importados diretamente em uma cena. A esse procedimento se dá o nome de importação de *assets*. *Assets* são artefatos que são importados de outras ferramentas de forma fácil e prática como um arrastar de mouse diretamente no projeto. A *engine* aceita diversos formatos de modelos 3D, sons,

Revista brasileira de iniciação científica
ISSN: 2359-232X

texturas e animações. Para o RoboGol virtual, tanto os robôs quanto as texturas foram importados como *assets*. Já a mesa, os gols, a pontuação, o placar de tempo e a bola foram criados a partir de elementos básicos encontrados no editor da UNITY.

O que garante o funcionamento adequado de um jogo é a movimentação dos elementos condicionados as respostas do ambiente 3D a essa movimentação (por exemplo, colisão). A *engine* utiliza internamente um motor de física da placa gráfica NVidia para efetuar simulações de comportamentos físicos dos objetos. Esse motor de física foi desenvolvido para ser executado utilizando o processamento de placas gráficas acarretando um melhor desempenho na velocidade e na renderização dos elementos 3D. Além da simulação física por trás desses elementos de colisão, os mesmos também são utilizados como gatilhos para geração de eventos que podem ser tratados e interpretados durante a execução da cena 3D.

Os scripts são linhas de códigos implementados em qualquer uma das três linguagens de programação suportados pela UNITY (JavaScript, C# ou Boo). Internamente, os scripts são executados por meio de uma versão modificada da biblioteca Mono (implementação de código aberto para o ".NET"). Scripts funcionam como qualquer outro componente e devem ser incluídos nos objetos que responderão aos algoritmos criados. Na UNITY a comunicação entre os objetos dentro de uma cena é realizada por meio de mensagens. Quando um objeto deseja interagir com outro, o primeiro realiza uma busca na cena pelo segundo e executa um chamado uma função de envio de mensagens. O segundo objeto recebe essa mensagem e responde a ela conforme sua codificação.

Requisitos e Modelagem

Com maior domínio sobre a ferramenta UNITY, iniciou-se a fase de levantamento de requisitos e da criação de artefatos UML. Os requisitos são regras de negócio (necessidades) ou mesmo solicitações às quais o sistema deve atender. Em outras palavras é o nome dado ao estudo das características que o sistema deverá ter para atender às necessidades e expectativas de um usuário final. Os requisitos foram levantados por meio de entrevistas com os jogadores do RoboGol e por meio do uso do equipamento físico. Os artefatos UML foram gerados e foi montada a documentação do projeto.

Revista brasileira de iniciação científica
ISSN: 2359-232X

A modelagem é a parte que se encarrega de *Transform*ar os resultados dos requisitos em um documento ou conjunto de documentos capazes de serem interpretados diretamente pelo programador. Para o desenvolvimento de jogos, um dos principais documentos criados é o diagrama de atividade. Esse diagrama mostra atividades sequenciais e paralelas de um processo. O diagrama é útil para realizar modelagem de processo de negócio ou fluxo de trabalho e de dados (Larman, 2007).

A Figura 2 representa o diagrama de atividade do gerenciamento do jogo 3D proposto. Esse primeiro diagrama de atividade produzido foi o do funcionamento geral do jogo, ou seja, todo o processo que não envolve o jogo e suas regras em si. O segundo diagrama (Figura 3) trata do jogo em si, como é o seu inicio, as ações possíveis de serem tomadas e o que determina seu termino.

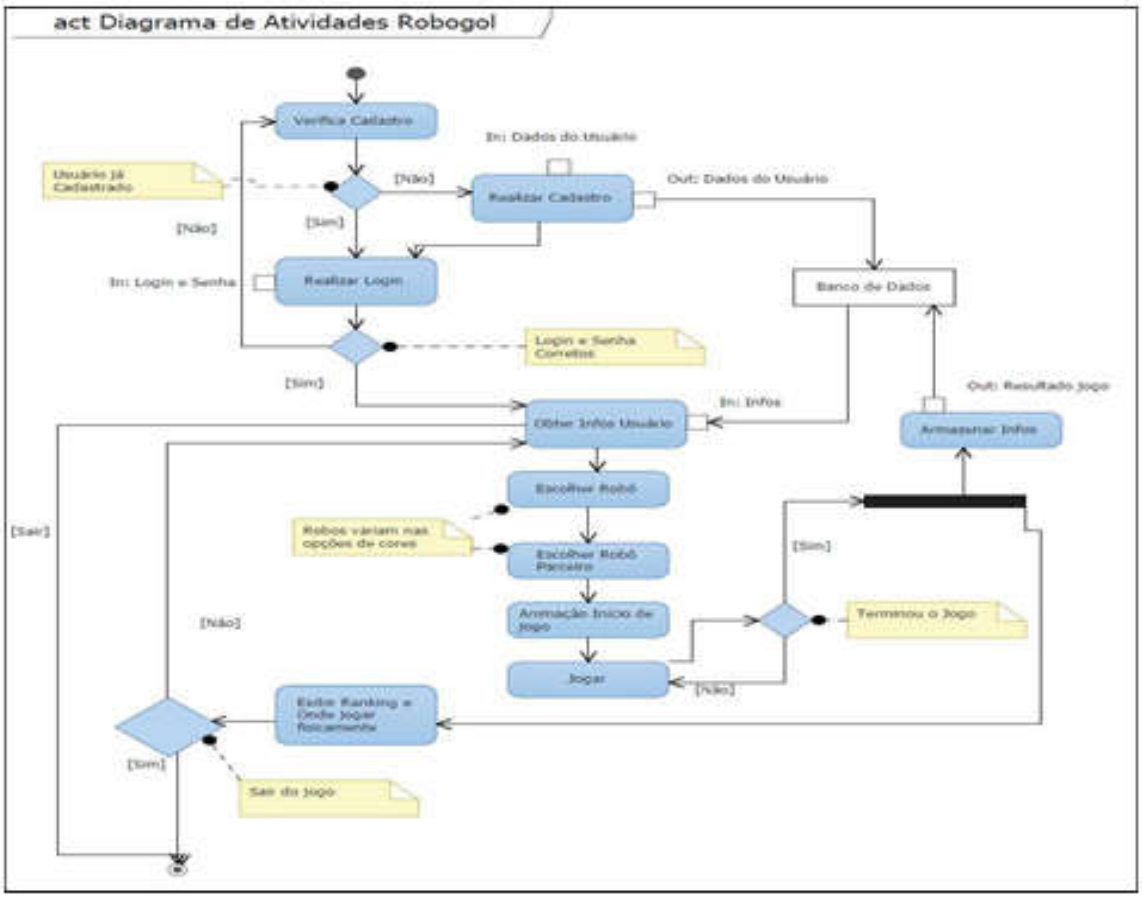

Figura 2 - Diagrama de atividade referente ao gerenciamento do RoboGol.

Revista brasileira de iniciação científica
ISSN: 2359-232X

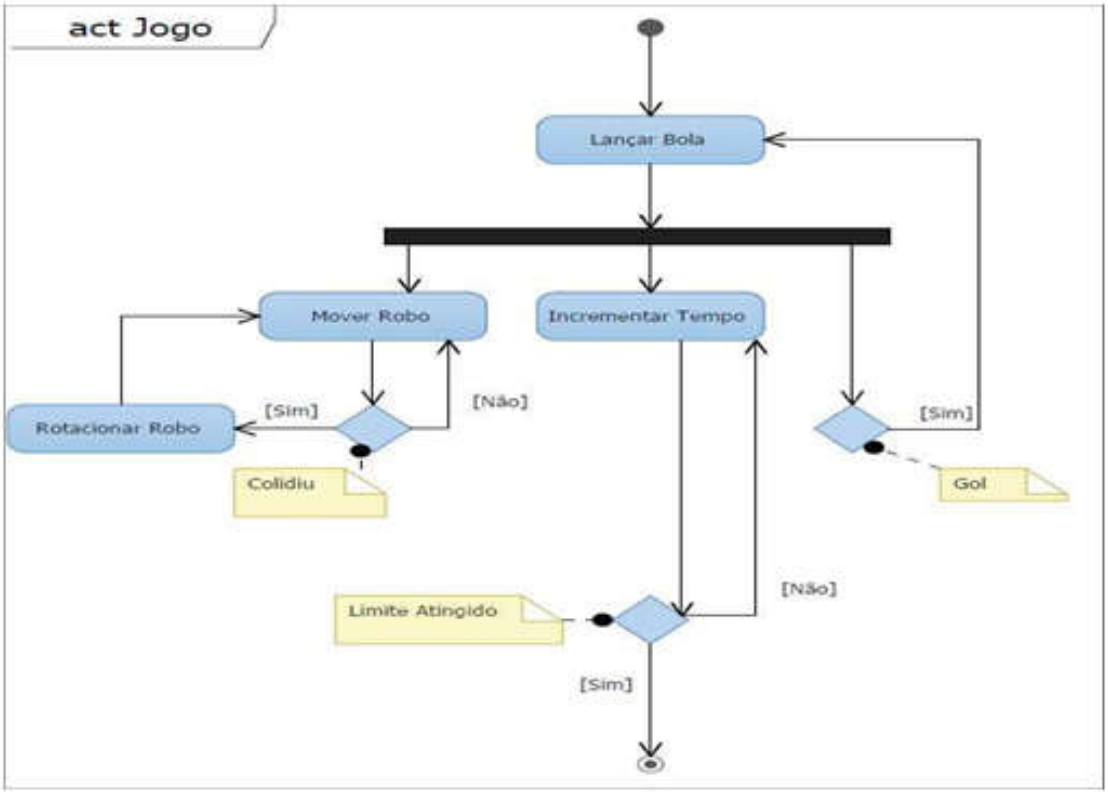

Figura 3 - Diagrama de atividade do jogo RoboGol virtual.

O diagrama de classes é fundamental para dividir os problemas complexos em problemas menores e mais fáceis de resolver. Esse diagrama facilita o entendimento do sistema como um todo e serve de base para a codificação do mesmo. O diagrama de classes do RoboGol virtual é apresentado na Figura 4.

Para a criação do RoboGol virtual foi necessário criar menus e telas de interação com o usuário. Foram desenhados esboços dessas telas (Figuras 5a, 5b e 6).

Revista brasileira de iniciação científica
ISSN: 2359-232X

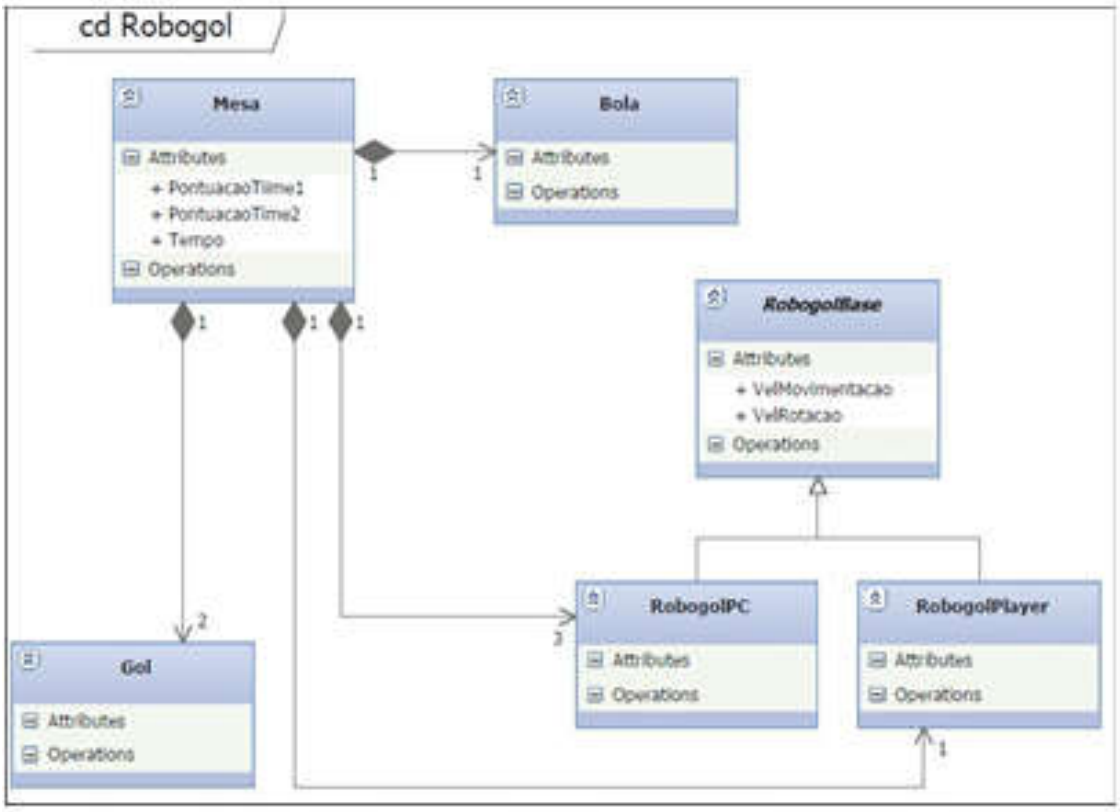

Figura 4 - Diagrama de classes do RoboGol.

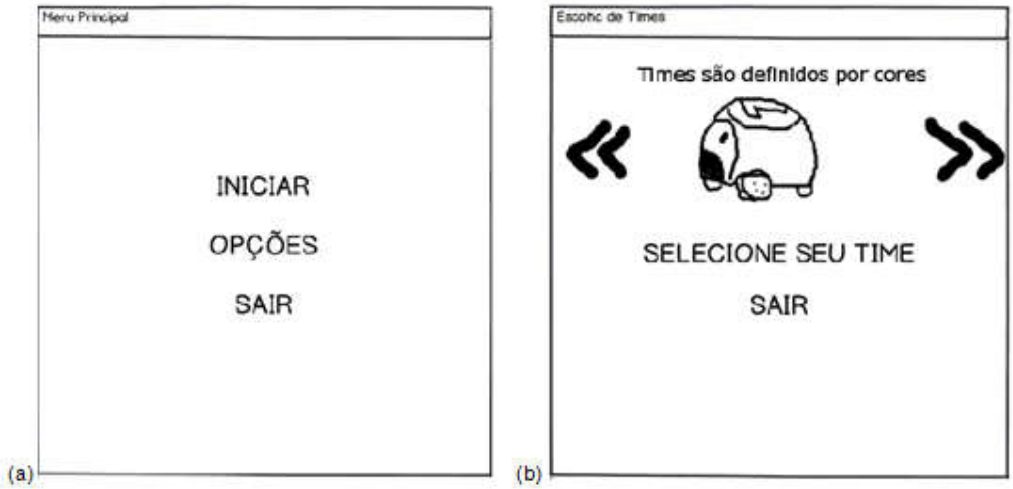

Figura 5 – (a) Esboço do menu principal. (b) Esboço da seleção de times.

Revista brasileira de iniciação científica
ISSN: 2359-232X

Classificação		P	J	V	E	D	Gols
1	Nome 1	45	40	3	36	1	126
2	Nome 2	36	40	1	33	6	107
3	Nome 3	10	6	2	4	0	15
4	Nome 4	7	6	1	4	1	26
5	Nome 5	7	3	2	1	0	4
6	Nome 6	5	6	0	5	1	17
7	Nome 7	4	4	0	4	0	20
8	Nome 8	4	3	1	1	1	7
9	Nome 9	3	3	0	3	0	5
10	Nome 10	2	2	0	2	0	3

Figura 6 – Esboço da tela da classificação.

Codificação

Para a codificação foi utilizada a técnica de orientação a objetos que possui como características principais a implementação de classes, herança, polimorfismo e encapsulamento. Diante disso, a linguagem de programação utilizada foi a C#. A codificação envolveu a criação de scripts para a captura de colisões entre a bola e os gols para efetuar a marcação de pontuação. Também foi realizada a codificação do contador de tempo e a finalização da partida quando o limite de tempo se esgota. Outro script importante realiza a movimentação do robô controlado pelo jogador humano e também o controlado automaticamente, isto é, quando o jogador joga sozinho contra o time controlado somente pelo sistema.

O *script* do robô controlado pelo jogador humano verifica qual a tecla está sendo pressionada no teclado e a partir dela define qual o movimento será realizado pelo robô. Caso a tecla pressionada seja a seta para cima o robô se movimentará para frente, caso seja a seta para baixo o robô se movimentara para trás. Caso seja a tecla da seta para esquerda, o robô efetuará uma rotação no próprio eixo para a esquerda e caso seja a seta para direita o robô rotacionará no próprio eixo para a direita. Para o robô controlado automaticamente o script

Revista brasileira de iniciação científica
ISSN: 2359-232X

executa um algoritmo simples. Quando o jogo inicia, os robôs do time automático obtém um valor randômico e a partir desse valor efetua a rotação para a esquerda ou para a direita, também, de forma randômica. Após essa rotação, cada robô verifica se irá para frente ou para trás e por quanto tempo. Caso um robô sofra uma colisão que não seja com a bola, ele repete o procedimento. Se a colisão for com o elemento bola, continua o movimento que já estava executando em direção ao gol adversário. Esse script de jogo automático foi chamado de "robô maluco".

A função do script da mesa é de computar o valor do gol para o time correto. Já o script da bola visa garantir que após a mesma entre no gol, esta seja relançada a partir do centro da mesa, como é o padrão do RoboGol físico.

Para realizar a publicação do jogo 3D em um servidor com acesso a Internet foi necessária escolher a opção "acessar" no menu *Build Settings* e escolher a plataforma *web*. Realizado este procedimento, o programa gera três arquivos: UnityObject.js, *Web*Player.html e *Web*Player.unity3d. No arquivo *WebPlayer.html* é possível realizar algumas edições como, por exemplo, o tamanho e posição da janela do jogo que ficará exibida no navegador, além das frases apresentadas durante o carregamento do jogo. São estes três arquivos que deverão ser anexados ao código do site no qual a aplicação ficará hospedada. A UNITY 3D assim como o Adobe Flash necessita que o usuário instale um player para seu funcionamento. Caso o usuário acesse a página *web* do jogo 3D e ainda não possua o *web* player, será aberta uma janela solicitando que o mesmo efetue a instalação do aplicativo. Nessa janela estará constando um link para o download do instalador.

Resultados Obtidos

A criação do menu principal foi realizada com a escolha de uma imagem de fundo lembrando o gramado de um campo de futebol sobreposto pelos botões com *labels*. Os botões são: iniciar, opções e sair. O botão "sair" finaliza o jogo e mostra o desempenho do jogador e sua classificação, o botão "opções" abre a tela de opções onde são apresentadas informações de como movimentar o jogador e opção de habilitar e desabilitar o som. Apesar desta versão não contar com som, o menu já foi preparado para tal função. E por último, o botão "iniciar" passa

Revista brasileira de iniciação científica
ISSN: 2359-232X

o jogo para a próxima tela que é responsável pela definição dos times por meio da escolha das cores dos robôs. Essa tela de escolha do time pode ser visualizada na Figura 7.

Figura 7 – Tela de jogo na seleção de times.

No exemplo exibido na Figura 7, o time do jogador seria o de cor laranja. No próximo passo, o jogador escolhe a coloração do time adversário, e logo após, o jogo inicia. O RoboGol virtual possui uma mesa, dois gols, quatro robôs e uma bola. A câmera de exibição foi posicionada simulando a posição que o jogador fica perante a mesa durante uma partida real. Para a iluminação foram posicionadas quatro luzes com os feixes em formato de cone, uma em cada canto da mesa dando a sensação de ambiente fechado. Para complementar o cenário foram posicionadas paredes para auxiliar na limitação visual que o jogador pode ter do ambiente 3D.

O jogo inicia com duração padrão de cinco minutos. Após o termino do tempo de jogo, o time que acumular mais gols é o vencedor. Ao término da partida, o jogo retorna ao menu inicial. Tanto os pontos efetuados (gols marcados) durante a partida quanto quem venceu ou perdeu são computados para compor o desempenho de cada jogador na classificação por pontos (ranking). São dados válidos para a composição da classificação os seguintes itens: pontos,

Revista brasileira de iniciação científica
ISSN: 2359-232X

jogos, vitórias, empates, derrotas e saldo de gols. A Figura 8 mostra uma cena do jogo 3D desenvolvido.

Figura 8 – Imagem do jogo 3D de uma partida em execução.

Dificuldades e limitações

Durante o processo de desenvolvimento do projeto algumas dificuldades foram observadas. No início das atividades a falta de conhecimento para o uso do software envolvido impediu que o trabalho fosse executado em um tempo adequado. Um exemplo envolveu o sistema de colisão da UNITY 3D que funciona de forma correta, porém seu entendimento demanda certo período de estudo de implementação. Esta falta de habilidade comprometeu no início, as regras de colisão que são necessárias para uma partida ser corretamente executada. Entre os problemas gerados houve a sobreposição de objetos e a demora na detecção da bola pelos robôs. O código teve que ser revisado várias vezes.

Como o projeto está relacionado a um ambiente 3D e é composta por diversos modelos tridimensionais, a falta de conhecimento em modelagem de objetos 3D foi uma grande limitação. Para contornar esse problema foram utilizados modelos 3D públicos encontrados na Internet, e os que não foram encontrados, como é o caso da mesa do jogo, foram criados com

Revista brasileira de iniciação científica
ISSN: 2359-232X

uma modelagem simplificada usando a própria ferramenta de desenvolvimento da *engine*. Outra limitação enfrentada foi à inclusão de sons no projeto. Neste caso não foi desenvolvido. De forma geral, um jogo digital é composto por uma equipe multidisciplinar para seu desenvolvimento. São programadores, designers, artistas gráficos, sonoplastas, entre outros. Diante disso, já era esperado que as principais dificuldades encontradas no desenvolvimento deste tipo de projeto seriam nas áreas não relacionadas diretamente com a programação.

Considerações finais

O desenvolvimento desse projeto contribui para o entendimento das fases que compõem a construção de um game 3D hospedado na Internet. A realização do trabalho em parceria com uma empresa proporcionou uma prática que ajudou a complementar a teoria aprendida em sala de aula adquirida durante a graduação. Além disso, permitiu aperfeiçoar os pontos de conhecimento técnico que ainda estavam com deficiência. É fundamental o contato com diferentes tecnologias, pois permite trabalhos em áreas que a Universidade não consegue abranger mesmo possuindo um curso extenso e amplo.

O projeto pode ser melhorado e dentre os requisitos adicionais para o seu aperfeiçoamento estão à conexão com as redes sociais para o envio dos resultados de partida (Facebook e Twitter) e a exibição do ranking de jogadores dentro da própria rede social (caso do Facebook), além da criação de um algoritmo inteligente para o jogo com um adversário automático. Outro ponto não contemplado foi à inclusão de sons. Como é de se esperar em um jogo 3D, o som é fundamental para melhorar a experiência do usuário no ambiente ao qual ele está interagindo.

Durante o desenvolvimento do jogo notou-se que seria mais atrativo aos usuários se o jogo 3D permitisse partidas entre dois a quatro jogadores em rede. Para que o jogo contemple essas partidas entre usuários em rede é necessário que o sistema aceite uma forma do usuário convidar outros jogadores e aguardar até que todos aceitem o convite para iniciar a partida. Dentro desse aspecto, o sistema deverá ser capaz de permitir que o usuário verifique a disponibilidade dos outros jogadores em tempo real para que possa enviar convites ou reservar um horário para iniciar a partida. Quando a partida der inicio é de se esperar que os jogadores possam se comunicar por voz entre eles.

Revista brasileira de iniciação científica
ISSN: 2359-232X

Referências

ROBOGOL, Sobre o produto. Disponível em: http://www.robogol.com.br/sobre/ [Visitado em maio de 2015].

UNITY 3D, Manual de Referência. Disponível em http://unity3d.com/support/documentation/Components/index.html [Visitado em maio de 2015].

UNITY 3Da, Tutorial. Disponível em http://unity3d.com/support/resources/tutorials [Visitado em maio de 2015].

MICROSOFT, IDE do Visual Studio para Windows. Disponível em http://www.microsoft.com/visualstudio/pt-br/products/2010-editions [Visitado em maio de 2015].

PRESSMAN, R. S., Engenharia de Software. 5ª edição, Rio de Janeiro: Editora McGraw-Hill, 2002.

LARMAN, C., Utilizando UML e padrões – Uma introdução à análise e ao projeto orientado a objetos e ao desenvolvimento iterativo. 3ª edição, São Paulo: Editora Bookman, 2007.

WATSON, K., Beginning C#: Programando. 1ª edição, Editora Makron Books, 2002.

GREENE, J. Use a Cabeça! C#. 1ª edição, Editora Alta Books, 2008.

FREEMAN, E. Use a Cabeça! Padrões de Projetos (Design Patterns), 1ª edição, Editora Alta Books, 2005.

BLACKMAN, S. Beginning 3D Game Development with Unity. 1ª edição, Editora Apress, 2011.

GOLDSTONE, W. Unity Game Development Essentials. 1ª edição, Editora Packt Publishing, 2009.

WITTAYABUNDIT, J. Unity 3 Game Development. 1ª edição, Editora Editora Packt Publishing, 2011.