Um Experimento de Arena de Robôs para o Ensino de Programação em Python

Cárlisson B. T. Galdino[1], **Alexandre A. Barbosa**[2],
Dyego Í. S. Ferreira[2], **Evandro B. Costa**[1], **Baldoino F. S. Neto**[1]

[1]Instituto de Computação – Universidade Federal de Alagoas (UFAL)
Campus A. C. Simões – Maceió, AL – Brazil

[2]Campus Arapiraca – Universidade Federal de alagoas (UFAL)
Arapiraca, AL – Brazil

carlisson@nti.ufal.br, alexandre146@gmail.com,

dyegoitallo@gmail.com, ebc.academico@gmail.com, baldoino@ic.ufal.br

Abstract. *This meta-paper describes the style to be used in articles and short papers for SBC conferences. For papers in English, you should add just an abstract while for the papers in Portuguese, we also ask for an abstract in Portuguese ("resumo"). In both cases, abstracts should not have more than 10 lines and must be in the first page of the paper.*

Resumo. *Este meta-artigo descreve o estilo a ser usado na confecção de artigos e resumos de artigos para publicação nos anais das conferências organizadas pela SBC. É solicitada a escrita de resumo e abstract apenas para os artigos escritos em português. Artigos em inglês deverão apresentar apenas abstract. Nos dois casos, o autor deve tomar cuidado para que o resumo (e o abstract) não ultrapassem 10 linhas cada, sendo que ambos devem estar na primeira página do artigo.*

1. Introdução

Cursos de Sistema de Informação, Ciência da Computação e afins enfrentam há anos uma dificuldade peculiar: como ensinar Programação aos alunos e obter deles dedicação e motivação?

Se por um lado Programação é a base do curso de Ciência da Computação, por outro é uma matéria que exige dedicação [Gomes and Mendes 2007]. Não basta estudar teoria, é preciso praticar e, para praticar, é preciso que os alunos se motivem.

Para despertar o interesse pela disciplina, foram propostas soluções, dentre elas o uso de Jogos Eletrônicos, no que [Chaudhary 2010] chama de DGBL - Digital Game-Based Learning.

DGBL, de maneira geral, pode ser feito com jogos criados sem o objetivo de serem usados como ferramentas de apoio ao processo educacional, ou com jogos feitos para esse fim, chamados de **serious games**. Para verificar se serious games ajudam no ensino de Programação, foi desenvolvido este trabalho.

2. Trabalhos Relacionados

Um experimento comparando a avaliação de alunos na forma mais tradicional com duas outras formas que envolviam um jogo foi feito por [de Jesus and Raabe]. Nele, concluiu-se que os alunos em dificuldade expostos ao jogo conseguiram progredir, deixando de ter dificuldades. Ademais, nenhuma outra diferença foi constatada.

Com abordagem mais simples [de Oliveira] criou o Arena, que trazia uma nova linguagem de programação inspirada em LOGO e com um modo de funcionamento dos robôs semelhante às regras de um Sistema Especialista. Em experimento com três grupos - um de controle com novatos, um grupo de novatos com uso da ferramenta Arena, um grupo de repetentes com a ferramenta -, foi registrado melhor desempenho nos dois grupos que utilizaram a ferramenta em relação ao de controle.

3. O Ambiente

Uma modalidade bastante popular de jogo eletrônico que auxilia no aprendizado de programação é um conceito que podemos chamar de ”arena de robôs”, do qual o mais popular é o Robocode, criado em 2001 e em atividade até hoje, feito em Java e oferecendo, desde 2010, suporte a .Net. Existem diversos jogos nesse estilo, que é relativamente antigo. O próprio Robocode foi criado com inspiração em outro projeto chamado Robot Battle [Larsen 2013] que, por sua vez, surgiu da inspiração no RobotWar [Schick]. Este foi criado em 1970[4] sendo, se não o mais antigo, um dos mais antigos do estilo e responsável por definir o gênero.

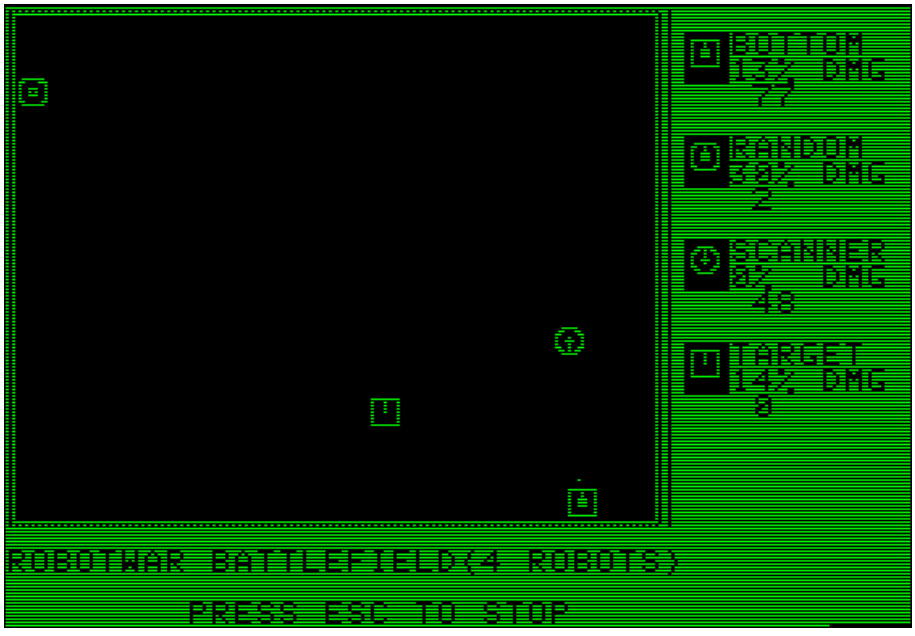


Figure 1. Tela do jogo RobotWar, de 1970, citado na sessão ??.

Em uma arena de robôs, temos uma área quadrada (a arena propriamente) onde robôs tentam se destruir. Cada robô é capaz de se locomover, disparar e girar o canhão. Somando-se a isso uma forma de perceber presença/localização dos oponentes, cada jogador participante deve escrever o código-fonte da ação que o robô deverá desempenhar na arena visando a destruir os demais.

3.1. Arenas de Robôs em Python

Para o nosso experimento, precisávamos de uma ferramenta de ”arena de robôs” que funcionasse para Python e que fosse facilmente utilizada pelos alunos. Foram encontradas e analisadas algumas soluções:

- **PythonRobocode** (PythonBots): Utiliza PyGame e apresenta uma interface bem acabada. Em testes, porém, não foi possível chegar ao momento dos combates. O aplicativo travou antes disso. [Macdonald 2011]
- **Python-Robocode**: Interface gráfica em PyQt4 oferece boas opções, mas muitas ainda não funcionais, o que tende a confundir o usuário. [turkishviking 2013]
- **pyRobocode**: Ainda não disponibilizou nenhum código [gorded 2007]
- **PyDroids**: Ainda não disponibilizaou nenhum código [caraciol]
- **pybotwar**: Usa PyQt4 e outras bibliotecas que não são facilmente encontradas/instaladas [Harr 2009]
- **Py Robocode**: Criado por Jurgis Pralgauskis e publicado inicialmente em 2009, não oferece funcionalidades via interface gráfica, resumindo-se a montar, quando executado, a arena com todos os robôs que estejam salvos em uma determinada pasta. Apesar de simples, o fato de funcionar sem problemas e sem maiores dependências (utiliza apenas a biblioteca Tkinter) pesou muito em sua escolha [Pralgauskis 2010]

Para construir um robô no Py Robocode, é necessário criar um arquivo de código-fonte em Python e representar o robô através de uma classe que herda seu comportamento da classe Bot. O único método que precisa ser implementado é o método tatics, que será evocado pelo py-robocode a cada iteração do combate.

De dentro desse método tatics é possível acessar informações sobre os robôs que estão lutando, o que inclui estado do canhão (se está atirando), ângulo e posicionamento, este já adaptado para um plano cartesiano, de modo que o eixo da Ordenada cresce de baixo para cima e não de cima para baixo, e o ponto central da arena sendo marcado como o (0, 0). O jogador precisa, com base nessas informações, definir a estratégia de ação que leve seu robô à vitória, eliminando os demais.

4. Metodologia

Definimos que, para este experimento, precisaríamos de alunos de Ciência da Computação que ainda não tivessem tido contato com Programação. Desta forma, proporíamos um Minicurso de Python à turma de 2014 de Ciência da Computação no Campus Arapiraca da UFAL.

No trato com esses alunos, planejamos o uso de dois grupos: um aprenderia Python sem contato com a ferramenta Py Robocode; o outro seria apresentado à ferramenta e seria convidado a participar de uma competição de robôs.

O Minicurso foi preparado com conhecimento básico de programação: atribuições, tipos de valor, expressões, estruturas de decisão e de repetição. O conteúdo foi distribuído pela manhã de três dias seguidos (segunda a quarta-feira) e na sexta seria aplicada uma avaliação dos conhecimentos adquiridos.

Além da avaliação final, os alunos responderam a um questionário no início do Minicurso para que pudéssemos mensurar seu conhecimento prévio. A inscrição dos

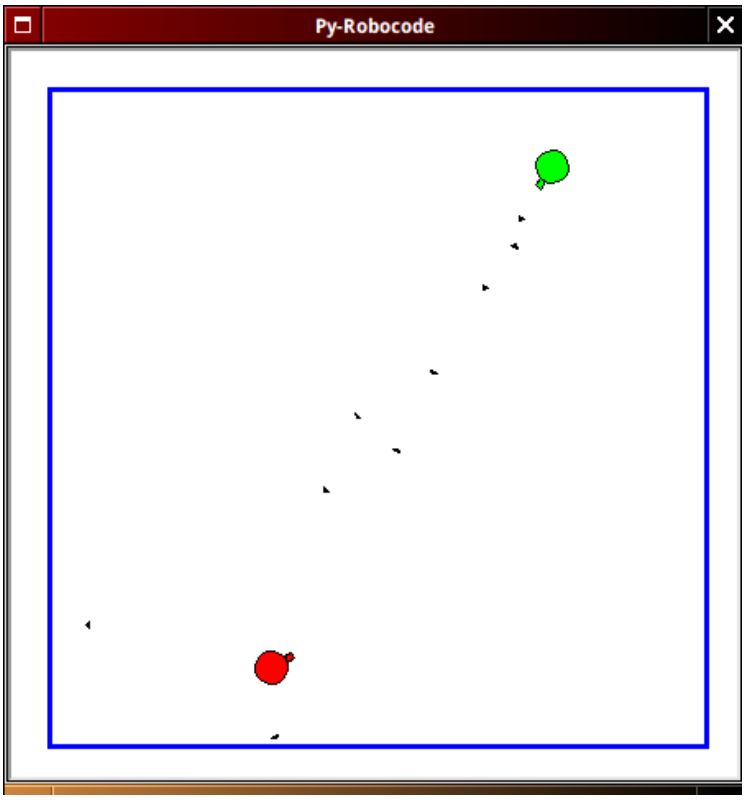


Figure 2. Py Robocode em execução.

alunos envolveu o aceite de um Termo de Consentimento, que informava aos alunos que eles estariam participando de um experimento que objetivava a melhoria do ensino de disciplinas de Programação.

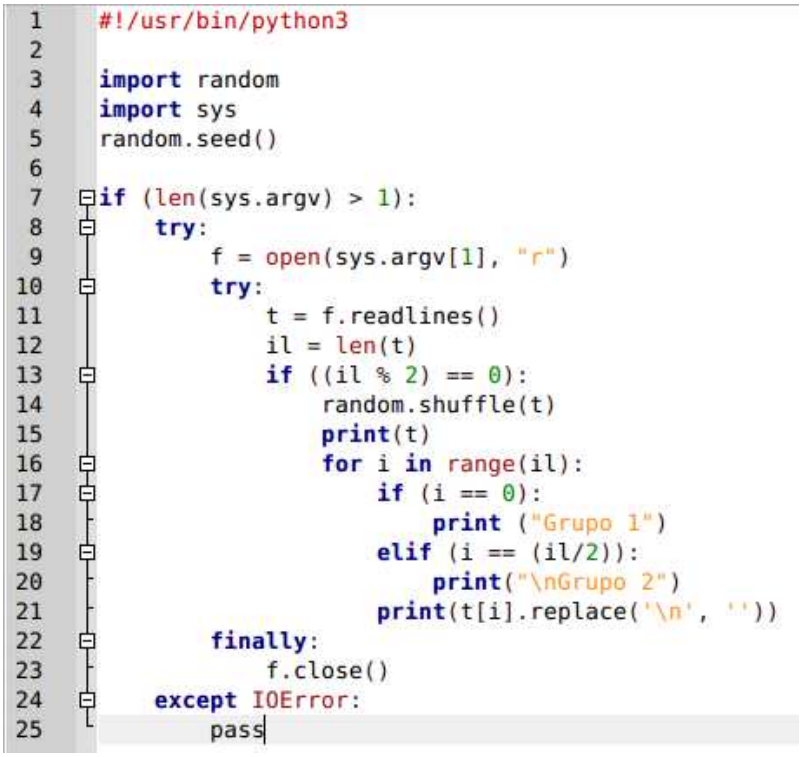
O minicurso seria aplicado duas vezes: primeiro para o grupo de controle; na semana seguinte, para o outro grupo. No caso do segundo grupo, o Py Robocode seria apresentado no terceiro dia, ao final da aula. Da mesma forma, a competição ocorreria após a avaliação, na sexta-feira.

5. Experimento

Recebemos 22 respostas ao convite para o Minicurso. Dividimos os 22 inscritos em duas turmas de 11. Para essa divisão, foi utilizado um script em Python (veja a Figura **??**), de maneira que fosse feita de maneira aleatória.

O questionário inicial continha três perguntas, sendo duas objetivas permitindo múltiplas opções e uma de duas opções mutuamente excludentes. A primeira pergunta era ”O que foi mais decisivo para você optar pelo curso de programação?”. A segunda: ”Possui algum conhecimento de programação?”. A terceira: ”Você gosta de jogos eletrônicos de algum tipo?”. As três perguntas apresentavam uma opção que permitia subjetividade, caso o aluno desejasse. O questionário foi respondido de forma anônima. Ficou registrado que 56% dos participantes (razão que, aproximadamente, se manteve nos dois grupos) escolheram cursar Ciência da Computação para poderem criar seus próprios programas. do que se pode deduzir que havia interesse concreto pelo minicurso oferecido.

Quanto a conhecimento prévio, como mostra a Tabela 2, 56% dos participantes



```
#!/usr/bin/python3

import random
import sys
random.seed()

if (len(sys.argv) > 1):
    try:
        f = open(sys.argv[1], "r")
        try:
            t = f.readlines()
            il = len(t)
            if ((il % 2) == 0):
                random.shuffle(t)
                print(t)
                for i in range(il):
                    if (i == 0):
                        print ("Grupo 1")
                    elif (i == (il/2)):
                        print("\nGrupo 2")
                    print(t[i].replace('\n', ''))
        finally:
            f.close()
    except IOError:
        pass
```

Figure 3. Algoritmo em Python utilizado para o sorteio das turmas.

Table 1. O que motivou os alunos a cursar Ciência da Computação

Questão 1	A	B	Respostas
1.a) Criar meus próprios programas	6	4	10
1.b) Montar uma empresa de tecnologia	3	3	6
1.c) Saber estruturar uma rede de computadores	0	1	1
1.d) Dar manutenção em computadores	0	1	1
1.e1) Identificação	1	1	2
1.e2) Trabalhar em área de computação que interesse	1	0	1
1.e3) Trabalhar em alguma grande empresa na área	0	1	1

não haviam tido contato com programação antes do minicurso. Apesar de a proporção ser razoavelmente boa, ela não se mostrou constante entre os grupos. já que no primeiro grupo essa razão sobe para 73%, enquanto no segundo cai para 17%. É bem possível que os 5 inscritos que não compareceram (todos do segundo grupo) aumentassem esse percentual. Várias linguagens de programação foram citadas pelos outros 44% dos alunos, sendo as mais conhecidas Java e Pascal (cada um por 18% dos alunos envolvidos no experimento). Cabe lembrar que o que foi questionado foi se havia algum conhecimento sobre programação, de modo que não devemos tratar como se houvesse ”domínio” sobre essas linguagens.

Ao indagarmos pelo estilo de jogo favorito, percebeu-se que todos marcaram pelo menos um estilo, no que se pode concluir que todos tem contato com jogos eletrônicos e gostam de jogar. Jogos de ação, incluindo tiro e plataforma, foram marcados por 71% dos participantes; seguidos pelos gêneros de Estratégia (incluindo RPG e simulação) e Esporte (incluindo corrida e luta), cada um com 35%. Curiosamente, apenas um dos

Table 2. Qual o conhecimento prévio dos alunos sobre Linguagens de Programação

Questão 2	A	B	Respostas
2.a1) Python	1	0	1
2.a2) JavaScript	1	0	1
2.a3)Java	2	1	3
2.a4) PHP	1	1	2
2.a5) Pascal	1	2	3
2.a6) C	0	2	2
2.a7) C++	0	1	1
2.a8) Não especificado	0	1	1
2.b) Não	8	1	9

alunos marcou jogos casuais.

Table 3. Preferência por estilos de jogos eletrônicos

Questão 3	A	B	Respostas
3.a) Ação, tiro e plataforma	8	4	12
3.b) Estratégia, RPG e simulação	3	3	6
3.c) Esporte, corrida e luta	3	3	6
3.d) Casual, musical e puzzle	0	1	1
3.e1) MOBA	1	0	1
3.e2) Aventura	1	0	1

O minicurso para a primeira turma ocorreu entre os dias 9 e 13 de junho deste, enquanto o da segunda turma ocorreu na semana seguinte, entre os dias 16 e 20.

Todos os alunos da primeira turma compareceram e participaram de todo o minicurso, já a segunda turma contou com participação de apenas 6 alunos, 2 dos quais não compareceram para a avaliação (Figura 4).

A apresentação do Py Robocode, com demonstração e explicação do código-fonte de um robô apresentado como modelo, levou cerca de 1h, sem registro de dúvidas entre os presentes. Todos os alunos copiaram o Py Robocode com o modelo para que se preparassem para a competição de sexta-feira. Após a avaliação, porém, não houve competição, visto que nenhum dos alunos havia escrito um.

A avaliação foi elaborada para que fosse possível constatar que os alunos haviam assimilado certos conhecimentos. Foram feitas duas provas diferentes, uma para cada turma, mas apresentando questões equivalentes em nível de complexidade e conhecimento testado. Por exemplo, enquanto na primeira prova tínhamos uma questão que pedia o cálculo do IMC, fornecidos peso e altura; a segunda prova pedia o cálculo da nota final, dadas a nota da prova final e a média. Para ambas, era fornecida na questão a expressão matemática capaz de calcular a resposta.

Dispondo de 3 horas, os alunos poderiam utilizar os computadores do Laboratório de Informática para testar soluções, mas tinham que escrever na prova a resposta para as 5 questões.

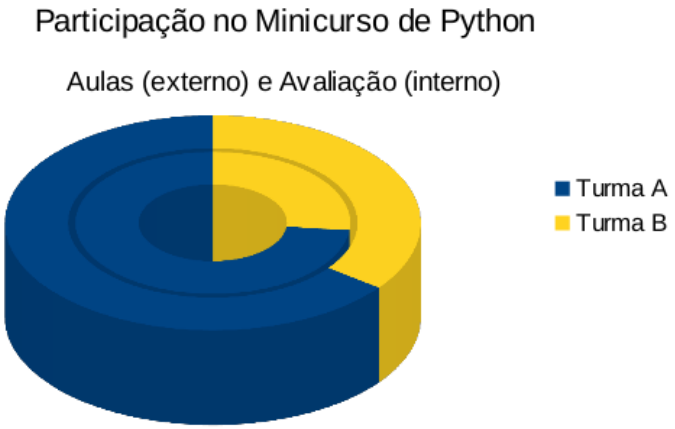


Figure 4. Participação dos alunos no Minicurso.

Na resposta, como mostra a Figura 5, cerca de 36,5% dos alunos da primeira turma conseguiram manipular corretamente strings, enquanto na segunda turma o índice foi de 75%. Nenhum aluno demonstrou conhecimento sobre listas. O uso de estruturas de repetição, terceiro e último critério investigado, foi satisfatório para 9% dos alunos da primeira turma, subindo para 25% da segunda turma.

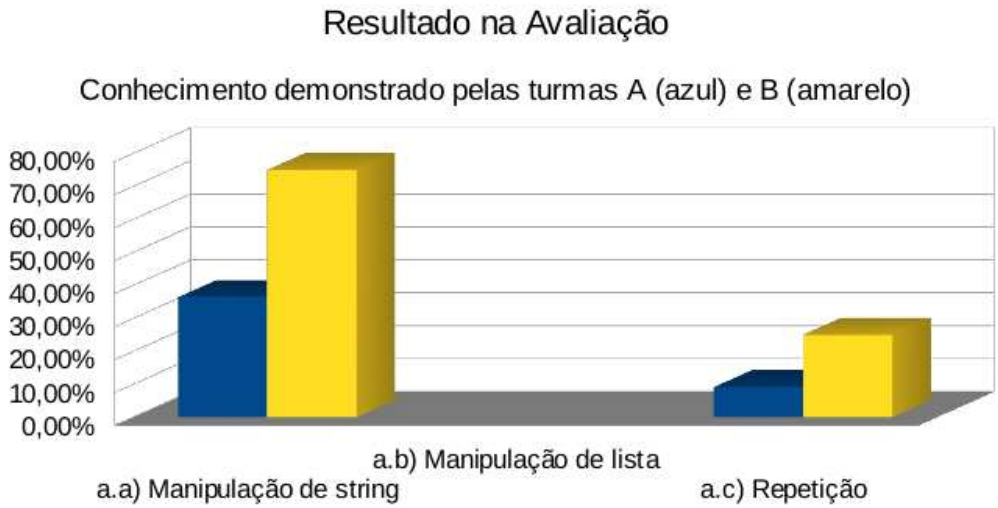


Figure 5. Comparativo do conhecimento demonstrado na avaliação.

Ao atentarmos para as notas na avaliação (Tabela 4, em média as notas da segunda turma (a que foi apresentada ao Py Robocode) foi maior. Tanto a maior nota quanto a menor da turma, porém, foram de algum aluno da primeira turma.

Assim, constatou-se um desempenho consideravelmente melhor na segunda turma, apoiando o uso de uma ferramenta de jogo voltada a Programação.

6. Discussão dos Resultados

O questionario inicial nos trouxe o perfil dos participantes do experimento de maneira simples. A primeira questão (Tabela 1) nos mostrou que havia interesse nos alunos pelo

Table 4. Resultado da avaliação com base nas notas

Avaliação	A	B	Respostas
Menor Nota	4,0	5,7	4,0
Nota Média	5,8	6,6	6,0
Nota Máxima	9,5	5,7	9,5

aprendizado de Programação. Dado que o Minicurso de Python era uma oferta de extensão, apenas os alunos interessados se inscreveram. Além de criar seus próprios programas (realidade para 60% dos participantes), os alunos também almejam criar suas próprias empresas de tecnologia (35%) ou mesmo trabalhar em uma grande empresa na área (como um dos participantes escreveu na opção subjetiva).

A segunda questão da pesquisa inicial (Tabela 2) mostrou que um número considerável de alunos já teve algum contato anterior com alguma linguagem de programação: cerca de 47%. O índice seria menos preocupante se fosse bem distribuído entre as turmas A e B. O fato de 73% dos participantes da primeira turma não ter esse conhecimento prévio e, na segunda, tal índice baixar para meros 17% se apresenta como o primeiro fato com potencial de comprometer os resultados deste trabalho, uma vez que não temos meio de medir qualitativamente esse conhecimento prévio e, assim, calcularmos o quanto influenciou no melhor desempenho da segunda turma.

Quanto aos estilos de jogo, Arena de Robôs pode ser classificado como um gênero de Estratégia e Simulação, com um pouco de Puzzle, o que o coloca, dentre as opções da terceira questão da pesquisa (Tabela 3), na opção **b**, tendo a opção **d** como secundária. Vendo desta forma, a solução Py Robocode teria maior apelo para 35% dos participantes. No caso da segunda turma, a razão de alunos com gosto por jogos de Estratégia, RPG e Simulação sobe para 50%, favorecendo o uso da ferramenta.

Outro aspecto preocupante para o experimento foi a baixa participação dos inscritos selecionados para a segunda turma: 55% participaram, contra 100% da primeira turma. Disto, tivemos que comparar resultados entre grupos de quantidades diferentes de membros. Mais que isso, desconhecendo o perfil dos outros 45%, é bem possível que fosse composto, em grande parte, por leigos, o que diminuiria a diferença de conhecimento prévio entre as duas turmas, caso houvessem participado.

Um segundo ponto que pode nos levar a questionar os resultados do experimento foi a falta de participação no torneio de Arena de Robôs. Apesar de a ferramenta ter sido explicada e fornecida para os participantes da segunda turma, nenhum robô foi construído. A alegação dos participantes foi que haveria uma avaliação importante do curso no período da tarde do mesmo dia, o que exigiu dedicação e estudo por parte deles, de modo a não terem tido tempo para se dedicarem aos robôs.

7. Conclusão e Trabalhos Futuros

A fim de confirmar os resultados obtivos, um novo experimento precisa ser realizado com outras turmas iniciais de cursos que apresentem Programação em sua grade.

Tentando tornar Programação algo ainda mais atrativo, sob a ótica do DGBL, há de ser desenvolvido um jogo digital mais complexo, capaz de divertir grande parte dos alunos. Para não comprometer a diversão, convém que este jogo não seja uma

ferramenta pedagógica, apenas fazendo uso de elementos de Programação. Segundo [Hunicke et al. 2010], 8 tipos de diversão podem ser encontradas em um jogo. A atividade de programar em muito se assemelha à solução de Puzzles - ou desafios -, que é o 4º tipo de diversão apresentado em seu trabalho. Com isso, um jogo pode ser divertido e equiparável a jogos digitais independentes apresentando Programação como apenas um de seus elementos. A viabilidade e utilidade de um jogo desse tipo precisa (e deve) ser posta a prova.

8. References

Bibliographic references must be unambiguous and uniform. We recommend giving the author names references in brackets.

The references must be listed using 12 point font size, with 6 points of space before each reference. The first line of each reference should not be indented, while the subsequent should be indented by 0.5 cm.

References

caraciol. pydroids. http://code.google.com/p/pydroids/. Acessado em Junho de 2014.

Chaudhary, A. G. (2010). Digital game-based learning - future of education?

de Jesus, E. A. and Raabe, A. L. A. Avaliação empírica da utilização de um jogo para auxiliar a aprendizagem de programação.

de Oliveira, R. N. Arena: Uma ferramenta auxiliar na disciplina ”introdução a computação” para programação de robôs.

Gomes, A. and Mendes, A. J. (2007). Learning to program - difficulties and solutions.

gorded (2007). pyrobocode. http://sourceforge.net/projects/pyrobocode/. Acessado em Junho de 2014.

Harr, L. (2009). pybotwar. https://code.google.com/p/pybotwar/. Acessado em Junho de 2014.

Hunicke, R., LeBlanc, M., and Zubek, R. (2010). Mda: A formal approach to game design and game research.

Larsen, F. N. (2013). Readme for robocode. http://robocode.sourceforge.net/docs/ReadMe.html. Acessado em Junho de 2014.

Macdonald, M. (2011). Python robocode. http://sourceforge.net/projects/pythonrobocode/. Acessado em Junho de 2014.

Pralgauskis, J. (2010). Py robocode. https://launchpad.net/py-robocode. Acessado em Junho de 2014.

Schick, B. A brief history. http://www.robotbattle.com/history.php. Acessado em Junho de 2014.

turkishviking (2013). Python-robocode. https://github.com/turkishviking/Python-Robocode. Acessado em Junho de 2014.