Pontifícia Universidade Católica
do Rio de Janeiro

Herminio Paucar Curasma

Uma ferramenta para a introdução à programação e pensamento computacional com motivação usando realidade virtual

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática da PUC-Rio.

Orientador: Prof. Alberto Barbosa Raposo

Rio de Janeiro
Outubro de 2017

Herminio Paucar Curasma

Uma ferramenta para a introdução à programação e pensamento computacional com motivação usando realidade virtual

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Alberto Barbosa Raposo
Orientador
Departamento de Informática – PUC-Rio

Prof. Bruno Feijó
Departamento de Informática – PUC-Rio

Greis Francy Mireya Silva Calpa
LNCC

Prof. Márcio da Silveira Carvalho
Coordenador Setorial do Centro Técnico Científico – PUC-Rio

Rio de Janeiro, 24 de outubro de 2017

Herminio Paucar Curasma

Graduou-se no 2013 em Engenharia de Software pela Universidade Nacional Maior de São Marcos (Lima, Perú). Começou o mestrado na primavera de 2015, durante o mestrado formou parte da equipe de Geofísica Computacional do Instituto Tecgraf.

Bibliographic data

Paucar Curasma Herminio

Uma ferramenta para a introdução à programação e pensamento computacional com motivação usando realidade virtual / Herminio Paucar Curasma; orientador: Prof. Alberto Barbosa Raposo. — Rio de Janeiro: PUC–Rio, Departamento de Informática, 2017.

v., 111 f: il. ; 29,7 cm

1. Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Incluí referências bibliográficas.

1.Pensamento computacional, 2. Realidade virtual, 3. Programação na educação, 4. Introdução à programação, 5. Programação e jogos, 6. Programação e realidade virtual, I. Barbosa Raposo, Alberto, II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática, III. Título.

CDD: 004

Agradecimentos

Primeiramente, a Deus, por tudo que me deu: minha família, todas as pessoas especiais, oportunidades e alegrias que colocou em minha vida.

A minha mãe Amacia Curasma pelo apoio em toda minha educação.

A meu orientador Alberto Raposo, pela ajuda que meu brindou durante toda a pesquisa.

A meus irmãos Maribel Paucar e Ronald Paucar pelo apoio moral e econômico que me brindaram para começar e terminar este grão logro.

A minha amiga Jessica Palomares e meus colegas de trabalho (conhecidos como as cobras da equipe I), pelos conselhos que facilitaram-me adaptar-me ao mestrado e a esta cidade maravilhosa que é o Rio de Janeiro.

A CAPES, à PUC-Rio e ao Tecgraf, pelos auxílios concedidos, sem os quais este trabalho não poderia ser realizado

Resumo

Paucar Curasma, Herminio; Barbosa Raposo, Alberto (Orientador). **Uma ferramenta para a introdução à programação e pensamento computacional com motivação usando realidade virtual.** Rio de Janeiro, 2017. 111p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

No nosso quotidiano ouvimos com frequência falar da importância das Tecnologias de Informação e Comunicação (TIC) pelos diversos atores sociais. A influência das TIC atravessa as diversas áreas da sociedade como: agricultura, serviços, comércio, indústria, investigação, entre outros. Se fizermos um raciocínio inverso será difícil nomearmos campos sociais que não sejam influenciados direta ou indiretamente pelas TIC. Além disso a demanda de trabalhadores em Computer Science e áreas relacionadas a STEM (Science, Technology, Engineering and Mathematics) está em aumento. É por isso mesmo que é importante que as crianças desde tenra idade se interessem pela tecnologia (Programação de computadores) e participem dela de uma forma divertida e lúdica. O presente trabalho propõe a criação de uma ferramenta de Realidade Virtual que permite que os estudantes aprendem conceitos básicos da programação e pensamento computacional tendo como finalidade que eles desfrutem da tecnologia e se sintam motivados em aprender mais. A ferramenta é uma Linguagem Visual de Programação. Os algoritmos são formados mediante a montagem de blocos-, resolvendo com isso um dos principais problemas dos estudantes que são os “erros de sintaxe”. Além disso a ferramenta traz consigo um conjunto de desafios ordenados por níveis, que têm como finalidade ensinar aos estudantes princípios básicos da programação e a lógica (programação sequencial, estrutura de dados repetitiva e condicional), onde em cada nível o aluno aprenderá as diferentes conceitos e comportamentos do pensamento computacional. Para as avaliações com os usuários se contou com a participação de 18 alunos com idades entre 12 e 15 anos provenientes de duas instituições públicas do Rio de Janeiro. Nestas avaliações considerou-se também medir a sensação de imersão mediante a Telepresença, Presença Social e Usabilidade.

Palavras-chave

Pensamento computacional; realidade virtual; programação na educação; introdução à programação; programação e jogos; programação e realidade virtual.

Abstract

Paucar Curasma, Herminio; Barbosa Raposo, Alberto (Advisor). **A tool for the introduction of programming and computational thinking with motivation using virtual reality.** Rio de Janeiro, 2017. 111p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Nowadays, we often hear about the importance of Information and Communication Technologies (ICT) by various social actors. The influence of ICT crosses the various areas of society as agriculture, services, trade, industry, research, among others. If we do an inverse reasoning, it will be difficult to name social fields that are not directly or indirectly influenced by ICTs. In addition, the demand for workers in Computer Science and areas related to the STEM (Science, Technology, Engineering, and Mathematics) is on the rise. That is why it is important to make the young person interested in technology (Computer programming) and participate in it in a fun and playful way. The present work proposes the creation of a Virtual Reality tool that allows students to learn basic concepts of programming and computational thinking with the purpose that they enjoy the technology and feel motivated to learn more. The tool is a Visual Programming Language; the algorithms are formed by block-assembly, thereby solving one of the students' main problems, which are "syntax errors". In addition, the tool brings with it a set of level-ordered challenges aimed at teaching students basic principles of programming and logic (sequential programming, repetitive and conditional data structure), where at each level the student will learn the different concepts and behaviors of computational thinking. For the evaluations with the users we counted on the participation of 18 students between 12 and 15 years old coming from two public institutions of Rio de Janeiro. In these evaluations it was also considered to measure the sensation of immersion through Telepresence, Social Presence and Usability.

Keywords

Computational thinking; virtual reality; programming in education; introduction to programming; programming and games; programming and virtual reality.

Sumário

Lista de figuras

Lista de tabelas

1
Introdução

Atualmente a demanda de trabalhadores em Computer Science e áreas relacionadas a STEM (Science, Technology, Engineering and Mathematics) está em aumento. Um estudo feito pelo departamento de comércio dos EUA tem uma previsão que a demanda de profissionais em computação vai continuar em aumento (BLS, 2017). É por isso mesmo importante que as crianças desde tenra idade se interessem pela tecnologia (programação de computadores) e participem dela de uma forma divertida e lúdica e em um futuro possam satisfazer a demanda de profissionais.

No nosso quotidiano ouvimos com frequência falar da importância das Tecnologias de Informação e Comunicações (TICs) pelos diversos atores sociais. A influência das TIC atravessa as diversas áreas da sociedade como: agricultura, serviços, comércio, indústria, investigação, entre outros. Se fizermos um raciocínio inverso será difícil nomearmos campos sociais que não sejam influenciados direta ou indiretamente pelas TIC. Alguns estudiosos afirmam que *"nos primeiros decênios do séc. XXI, mais de 80% dos seres Humanos terão acesso ao ciberespaço e se servirão dele quotidianamente"* e que *"as atividades de pesquisa, aprendizagem e de lazer serão virtuais ou comandadas pela economia virtual"* (Levy, P., 1993). A educação não pode ficar alheia a estas evidências.

A par do investimento em tecnologia nas escolas, tem sido proporcionada também formação básica em TIC para os professores. *"A integração efetiva das TIC na educação tem de ir para além de simplesmente melhorar a eficiência ou acelerar as práticas atuais. As TIC ainda não tiveram um impacto significante nas abordagens de aprendizagem e de ensino; estão de fato ainda subaproveitadas"* (Van den Branden et al., 2009). Constata-se, no dia-a-dia das escolas, que alguns profissionais da educação são defensores teóricos da integração das TIC nos currículos, todavia as suas práticas educativas andam

muitas vezes defasadas de uma real integração das TIC no processo de ensino-aprendizagem.

Uma das dificuldades que as pessoas têm quando começam a aprender a programação são os erros de sintaxe. Em um estudo feito por Lahtinen et al. (2005) e Robins et al. (2003) nos mostram alguns problemas relevantes no processo de aprendizagem e relacionam a sintaxe das linguagens de programação como um desses problemas.

Essa observação é reafirmada no trabalho de Paul Denny et al. (2011). Nele, Denny e os outros autores buscam compreender a barreira da sintaxe para os programadores iniciantes. Com o objetivo de conduzir a pesquisa, formularam as seguintes perguntas: *"P1) Quão frequentes são os erros de sintaxe nas submissões de exercícios? Essa frequência está relacionada ao desempenho dos alunos no resultado final do curso? "* e *"P2) Até que ponto os erros de sintaxe atrapalham os estudantes no que diz respeito a receber feedback sobre a lógica de seus programas?"*. No total, participaram da pesquisa 330 alunos. Os exercícios que resolveram lhes foram apresentados como cabeçalhos de métodos da linguagem Java. Assim, a tarefa era preencher os métodos adequadamente. Como tarefa, cada estudante deveria criar 1 exercício e responder a pelo menos 10 num período de 4 semanas. No final das atividades, os autores classificaram os alunos participantes num ranking baseado no desempenho final na disciplina de Introdução à Programação. A partir daí, dividiram os alunos já classificados em quatro quartis (Q1, Q2, Q3 e Q4) e analisaram os dados para responder às perguntas de pesquisa P1 e P2.

Os autores (Denny et al., 2011) verificaram que a frequência de submissões com problemas de sintaxe é de aproximadamente 50% para Q1, cerca de 60% para Q2, 65% para Q3 e 73% para Q4. Mesmo sendo o menor valor, o número de problemas relacionados a sintaxe em Q1 era muito maior do que os pesquisadores esperavam. Ao analisar os dados para encontrar uma resposta a P2, os autores perceberam uma clara tendência a falha no fim do curso para aqueles alunos que não conseguiram compilar e executar seus códigos durante o experimento.

O desenvolvimento da tecnologia informática tem causado um enorme impacto na educação, especialmente quando se trata da tecnologia da Realidade Virtual (RV), que está mudando muito as idéias e os métodos de educação. Esta tecnologia tem como base à: visualização, interação homem-máquina e simulação.

Uma das características principais da RV é a imersão no mundo virtual, ele tem dois ingredientes-chave de experiências inesquecíveis como *"eu estava lá"* e *"eu senti"*(Parmar, D. 2016).

Um estudo feito por T. Nadan et al (2011) onde eles impartiram uma aula usando dois ambientes uma de Realidade Virtual e a outra um laboratório tradicional, nesta aula eles ensinaram conceitos de engenharia, os autores concluíram que a RV é uma experiência de aprendizagem mais memorável do que demonstrações baseadas em laboratório.

Com a descrição do estudo mencionado anteriormente, sobre o problema que os estudantes têm com a sintaxe das linguagens de programação, este trabalho propõe uma ferramenta de realidade que desenvolve os algoritmos com a montagem dos blocos (Drag and drop) e não com a escritura de código fonte, o que permitirá que o aluno se concentre na lógica de programação e no funcionamento das estruturas destinadas ao desenvolvimento dos algoritmos (laços de repetição, estrutura de decisão, entrada/saída de dados, etc.), deixando de lado os erros de sintaxes que poderia atrapalhar sua aprendizagem.

1.1. Objetivos

Desenvolver uma ferramenta que permita ensinar de uma forma introdutória a programação e o pensamento computacional aos estudantes (crianças e jovens), e que elas participem de uma forma divertida e lúdica.

Para isso teremos os objetivos secundários:

a) Desenvolver uma ferramenta de Realidade Virtual (Compilador Visual de Programação) que funcione com a montagem de blocos. Cada bloco apresenta o aluno a uma determinada instrução de código, que pode ser: funções pré-definidas, estruturas de controle repetitivas e estruturas de controle condicional.

b) Desenvolver 10 desafios os quais servirão para que o estudante aprenda em cada desafio um novo conceito e uma abordagem.

c) Usar a tecnologia de imersão que provê o Oculus Rift e o Microsoft Kinect para visualizar a execução do Script que dá solução para cada desafio.

d) Realizar a avaliação da ferramenta em duas escolas do Rio de Janeiro, tendo como um mínimo de 18 participantes que pertençam ao ensino fundamental. Isto para conhecer os efeitos da Realidade Virtual no comportamento dos estudantes.
e) Analisar os resultados das avaliações feitas em 3 fases pré-teste, teste e pós-teste.

1.2. Organização do Trabalho

O texto desta dissertação está organizado da seguinte forma. O capítulo 2 fornece os conceitos sobre o "pensamento computacional", "Linguagem Visual de Programação" e as tecnologias usadas como Oculus Rift e Microsoft Kinect. O capítulo 3 fornece um resumo dos trabalhos relacionados com o enfoque principal em descrever as ferramentas que foram criadas para o ensino da programação e pensamento computacional; estas ferramentas foram agrupadas em 2D e 3D. Além disso se descreveram como o pensamento computacional foi integrado na educação para o ensino de outras ciências. O capítulo 4 fornece a metodologia de desenvolvimento da solução, neste capítulo descreve-se a metodologia usada e detalha-se seus respectivos componentes. O capítulo 5 fornece a avaliação da ferramenta, neste capítulo descreve-se a metodologia de avaliação científica (Quantitativa e Qualitativa) e os resultados da avaliação. Finalmente, o capítulo 6 fornece as conclusões sobre este trabalho e propostas de trabalhos futuros.

2
Conceitos e Tecnologias utilizados

2.1.
Visual programming language (VPL)

As linguagens visuais de programação (VPLs) permitem que programas sejam “desenhados” usando blocos, onde cada bloco é unido a outro para formar um script de código. VPLs são normalmente utilizadas para fins educacionais, multimídia, jogos de vídeo, desenvolvimento de sistemas, simulação, automação e armazenamento de dados. Por exemplo, Scratch (Scratch, 2017) é uma plataforma desenvolvida pelo Instituto Tecnológico de Massachusetts, projetada para as crianças de etapa escolar; Pure Data (PD) (Pure Data, 2017) está projetada para a criação de multimídia interativa e música de computador; Unreal Engine 4 (Unreal Engine, 2017) usa "Blueprints" para programar jogos de vídeo; VisSim (VisSim, 2017) Permite que o usuário faça complexos modelos matemáticos de forma mais inteligente e rápida, enquanto os executa em tempo real; CiMPLE (CiMPLE, 2017) é usada para ensino da automatização através da robótica. Por linguagem visual queremos dizer o uso sistemático de expressões visuais para transmitir o significado de algo e essas expressões são conhecidas como sentenças visuais. A linguagem de programação visual é uma distribuição espacial em ícones bidimensionais (gráficos) que constituem uma sentença visual.

As linguagens usam ícones generalizados. Um ícone generalizado é um objeto com uma representação double, uma parte lógica (significado) e uma parte física (imagem) (Chang, 1990). Uma visualização é formada por representações gráficas que ilustram um dado, um programa, a estrutura de um sistema complexo, ou o comportamento dinâmico de um sistema complexo. Finalmente, um sistema de programação visual é um sistema informático que suporta programação visual e visualização (Chang, 1990).

2.2.
Pensamento Computacional - Computational Thinking

O pensamento computacional é foco de vários estudos e relatórios nos últimos anos (Guzdial 2008, Qualls & Sherrell, 2010). (Wing, 2006) o define como um conjunto de habilidades intelectuais e de raciocínio que afirma como as pessoas interagem e aprendem a pensar através da linguagem de programação. Em outras palavras, pensar computacionalmente envolve o uso de métodos, linguagem e sistemas de ciência da computação (CS) para resolver problemas em qualquer disciplina.

Muitos autores afirmam que o pensamento computacional é vagamente definido e uma definição clara é necessária para usar essa construção para obter informações sobre problemas (Guzdial, 2008; Dennings, 2009). Por exemplo, (Wing J. M., 2006) argumenta que o Pensamento Computacional incorpora todas as habilidades críticas e que envolve a resolução de problemas com o pensamento matemático e de engenharia.

No entanto, um estudo recente que investigou a importância das habilidades que desenvolvem com o pensamento computacional revela que o pensamento matemático e de engenharia não são necessariamente uma característica principal do pensamento computacional, porque pensamento computacional também pode ocorrer espontaneamente (Ater-Kranov et al., 2010).

O trabalho recente neste campo examinou as categorias de pensamento computacional, resumindo a lógica derivada da literatura e, de acordo com esta pesquisa, nenhuma dessas categorias inclui pensamento matemático e de engenharia (Berland & Lee, 2011). Embora a evidência empírica esteja atualmente ausente da literatura, as categorias de pensamento computacional que os acadêmicos concordam são: *lógica condicional, algoritmos de construção, depuração, simulação e computação distribuída* Wing (2006, 2008) (Ater-Kranov et al., 2010) (Berland & Lee, 2011). A *lógica condicional* é o bloco de construção do pensamento computacional e refere-se às consequências locais do valor verdadeiro/falso de uma determinada sentença. Os *algoritmos de construção* contêm um conjunto de lógica condicional e apresenta instruções para resolver um problema complexo em uma abordagem passo a passo. A *depuração* se refere ao ato de determinar problemas em um algoritmo, a *simulação* estabelece o

modelado e a implementação do algoritmo como um banco de testes para identificar quais circunstâncias e abstrações deve-se considerar. Finalmente, a *computação distribuída* refere-se ao aspecto social do pensamento computacional e envolve múltiplas partes ao desenvolver abstrações.

Muitos autores assinalam que o pensamento computacional não é sinônimo de programação (Wing, 2006; Guzdial, 2008; Repenning, Webb & Ioannidou, 2010). No entanto, uma pesquisa revelou que a maioria dos professores do ensino médio acredita que o pensamento computacional é idêntico à programação (Blum & Cortina, 2007). Portanto, neste ponto, é crucial diferenciar uma ferramenta de programação de uma ferramenta de pensamento computacional. Uma ferramenta de programação deve apoiar os alunos em escrever programas, fornecendo feedback sobre erros de sintaxe, implementação de métodos e lógica de programação (Haden & Mann, 2003). Da mesma forma, uma ferramenta de pensamento computacional deve oferecer um mapeamento simples entre um problema e suas soluções alternativas usando o feedback relevante e um contexto familiar para os alunos. Por um lado, temos ferramentas de programação em que as atividades geralmente envolvem a escrita de código de programação excessivo para aprender a estrutura de programação e produzir resultados eficientes, enquanto que, por outro lado, uma ferramenta de pensamento computacional pode permitir o desenvolvimento de soluções simples para desafios de ciências da computação com pouco ou nenhum fundamento de programação. Recentemente, pesquisadores neste campo enfatizaram que as ferramentas podem tornar a pensamento computacional mais acessível para todos (Qualls & Sherrell 2010; Kazimoglu et al., 2011).

Alguns outros autores consideram dois componentes para definir o pensamento computacional, estes são *conceitos* e *comportamentos* “Concepts and Approaches” onde os *Concepts* estão compostos por: *Logic, Algorithms, Decomposition, Patterns, Abstraction* e *Evaluation.* Onde a, *Logic* refere-se ao raciocínio lógico que ajuda a explicar porque algo acontece. A *Algorithms* refere-se ao conjunto de instruções o um conjunto de regras ordenadas para fazer algo. A *Decomposition* refere-se ao processo de quebrar um problema em pequenas partes, isto ajuda a resolver complexos problemas e gerenciar grandes projetos. A *Patterns* refere-se a encontrar patrões de repetição com os quais nós podemos fazer predições, criar regras de comportamento e resolver problemas gerais. A

Abstraction refere-se a simplificar as coisas, identificando o que é importante sem se preocupar muito com o detalhe. A abstração nos permite gerenciar a complexidade. A *Evaluation* refere-se a fazer julgamentos, de forma objetiva e sistemática, sempre que possível (Barefootcas, 2017).

Os *Approaches* estão compostos por: *Tinkering, Creating, Debugging, Persevering* e *Collaborating.* Onde o *Tinkering* refere-se à atividade de tentar tudo. é a primeira fase para aprender alguma coisa, consiste na exploração e experimentação, por exemplo quando uma criança de 6 anos começa a interagir por vez primeira com o computador, ele começa a mexer o mouse e dá-se conta que o cursor da janela também está em movimentação. A *Creating* refere-se, ao planejamento, elaboração e avaliação de animações, jogos ou robots. O *Debugging* refere-se ao processo de encontrar e corrigir os erros (*Bugs*), algumas vezes este processo pode levar muito mais tempo do que escrever o código. O *Persevering* refere-se a nunca desistir, ser determinado, resiliente e tenaz. A *Collaborating* refere-se à atividade de trabalhar com outras pessoas para garantir o melhor resultado (Computational thinking - Barefootcas, 2017).

No contexto desta discussão, o presente trabalho desenvolveu uma ferramenta visual de programação que permita ensinar de uma forma introdutória a programação e o pensamento computacional aos estudantes (crianças e jovens) , e que elas participem de uma forma divertida e lúdica através da interação e imersão na ferramenta. Durante a avaliação desta ferramenta os participantes farão as seguintes atividades do pensamento computacional.

- Criar e aplicar algoritmos para dar solução aos diferentes desafios.
- Avaliar um algoritmo e explicar o apropriado critério usado.
- Depurar os algoritmos e detectar erros lógicos.
- Simular os algoritmos em um ambiente virtual e observar o comportamento.

2.3. Oculus Rift

É um dispositivo usado na cabeça do usuário, que tem uma tela na frente dos olhos que exibe duas imagens lado a lado, uma para cada olho. Um par de lentes ópticas é colocado no topo da tela para possibilitar o foco nas imagens tão de perto e cobrindo todo o campo de visão do portador. As imagens exibidas são

muito semelhantes, mas têm diferentes perspectivas e assim é possível a visualização de uma imagem 3D (Oculus Rift, 2014). Figura 2.1 vemos a foto deste dispositivo.

Figura 2.1 Dispositivo versão DK2 (Oculus Rift, 2014)

O dispositivo possui as seguintes configurações:

- A resolução total da tela é 1280 x 800, mas como é dividida, então a resolução fica 640 x 800 por olho;
- O tamanho total da tela na horizontal é de ~17,78 cm (7 inches);
- O tempo de persistência é aproximadamente 3ms;
- O tempo de atualização de dados (refresh rate) é 60Hz;
- O campo de visão (field of view) nominal é 110°.

Os óculos têm 3 sensores incorporados que monitoram movimentos da cabeça do utilizador:

- O magnetômetro, que mede campos magnéticos, e assim torna possível determinar a orientação absoluta;
- O acelerômetro, que mede acelerações, e é útil para medir alterações na velocidade;
- O giroscópio, que mede mudanças de orientação ou mudanças na velocidade de rotação (giro taxa).

A razão pela qual estes sensores são combinados é porque o magnetômetro tem pouca precisão quando movimentado rapidamente, e o giroscópio requer uma orientação conhecida no início, mas reage com alta precisão a alterações; porém acumula erro ao longo do tempo. Dessa forma é possível rastrear as rotações da cabeça nos eixos X, Y e Z, como mostrado na Figura 2.2.

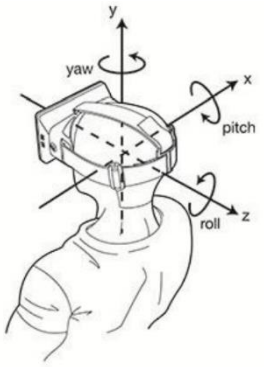


Figura 2.2 Eixos de rotação (Musa, 2013)

2.4. Microsoft Kinect

O Kinect foi desenvolvido pela Microsoft Research em parceria com a empresa israelense PrimeSense, para o console de videogame Xbox 360, Xbox One, também para o Windows. A primeira versão do sensor foi lançada para o Xbox 360 em 20 de novembro de 2010, e a segunda, para o Xbox One, foi apresentada em 22 de novembro de 2013. A primeira versão do Kinect para o Windows foi disponibilizada para os desenvolvedores em 1 de novembro de 2011, enquanto a segunda versão foi liberada em 15 de julho de 2014.

Os componentes do sensor do Kinect na Figura 2.3 incluem uma câmera RGB (Color sensor) que armazena os dados em três canais: um canal com resolução de 1280x960 a 12 quadros por segundo (fps), um canal com resolução de 640x480 a 30 fps, e um canal YUV com resolução de 640x480 a 15 fps. Além disso, o sistema possui um emissor (IR Emitter) e um sensor de infravermelho (IR Depth Sensor). O emissor lança feixes de luz infravermelha que são refletidos e detectados pelo sensor de profundidade. Os feixes refletidos são convertidos em dados de profundidade que informam a distância entre o objeto e o sensor. O Kinect dispõe, ainda, de um conjunto com quatro microfones (Microphone Array) para captar o som, e um motor de inclinação (Tilt Motor) que determina a orientação do sensor (Microsoft Kinect, 2017).

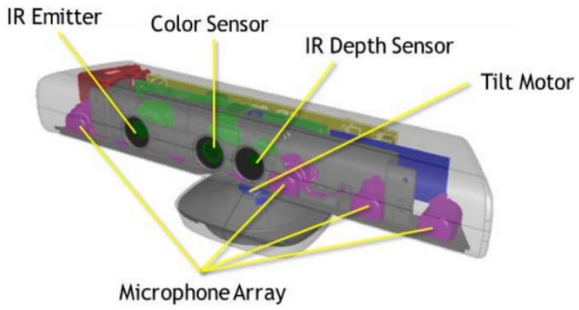


Figura 2.3 Componentes do sensor Kinect

O uso do Microsoft Kinect para a educação possibilita ao estudante a interagir com os objetos do sistema onde ele é inserido, e proporciona-lhe um ambiente rico em experiência, permitindo-lhe trabalhar com seus sentidos visual, auditivo e sistema motor.

3
Trabalhos Relacionados

Este capítulo descreve as ferramentas mais comuns que foram criadas para o ensino da programação e pensamento computacional e que são focadas nos estudantes de ensino fundamental. Estas ferramentas serão divididas em 2 grupos: 2D e 3D. No grupo de 2D se explicará as ferramentas mais importantes, cuja interface de desenvolvimento está em duas dimensões, e para o grupo 3D, em três dimensões. Também discutiu-se como estas ferramentas são aplicadas para o ensino do pensamento computacional, e como elas são aplicadas nas escolas. Além disso discute-se como o pensamento computacional é aplicado para ensino de outras matérias.

3.1.
Ferramentas em 2D

Estas ferramentas oferecem uma interface de saída 2D. Elas são conhecidas por usar imagens que representam os blocos e têm como finalidade executar animações, videogames, movimentações de robôs, etc. Algumas delas trabalham com componentes externos, como robôs. As ferramentas mais conhecidas são apresentadas a seguir.

3.1.1.
Scratch

Scratch é uma linguagem de programação visual projetada pelo MIT (Massachusetts Institute of Technology) Media Lab, lançada em 2005. O software ajuda aos mais jovens a aprender a pensar criativamente, raciocinar sistematicamente, e trabalhar colaborativamente através do desenho de seus próprias histórias interativas, jogos e animações. Estes projetos podem ser compartilhados em uma comunidade online, a qual até o momento tem cerca de 17 milhões de usuários registrados e quase 20 milhões de projetos (Scratch, 2017). Scratch tem sido usado de variadas formas para ensinar conceitos de pensamento computacional.

No estudo feito por Boechler et al. (2014), desenvolveu-se um curso que tinha como objetivo principal o treinamento dos professores, sobre como eles deveriam incorporar o pensamento computacional em suas aulas. Os autores se focaram no desenvolvimento de jogos usando Scratch. Posteriormente, eles testaram mediante algumas avaliações que consideram: número de scripts, número de blocos, número de variáveis e nível de complexidade do algoritmo. Dasgupta et. al. (2016) observaram que os usuários que usam Scratch online permitem que outros usuários possam aprender e realizar algumas modificações na aplicação.

DISSECT (Nesiba, 2015) (Arraki, 2014) é uma iniciativa da Universidade do Estado do Novo México, que incorpora o pensamento computacional no 6° ano do ensino. Eles usam diferentes técnicas e ferramentas, ressaltando o Scratch, usado para reforçar as habilidades e conceitos que foram aprendidos em outras matérias. O software Scratch foi recebido de forma positiva pelos estudantes. Posteriormente, fizeram as avaliações com escolas do mesmo nível (6°) e os pesquisadores descobriram que os alunos que tiveram aulas de Scratch entendiam melhor alguns conceitos mais avançados. Também verificaram que estes alunos estavam muito interessados e motivados a conhecer mais tópicos de ciências da computação.

No estudo feito por Grover et. al. (2014, 2016) desenvolveu-se um Currículo chamado FACT, o qual eles ensinam a estudantes de ensino fundamental. Eles usaram Scratch como sua principal ferramenta e ensinaram conceitos como loops, variáveis, entrada de dados, algoritmos e condicionais. Os pesquisadores descobriram que seu curso, que durou 24 horas durante seis semanas não deu resultados estatisticamente melhores do que o curso israelense de 60 horas dado ao longo de um ano, no entanto, existem outros fatores contribuintes, como os estudantes israelenses terem sido escolhidos dentre os melhores de sua turma.

No estudo feito por Imberman et al. (2014) foi reportado que uma fundação do Google chamada CS4HS inclui ferramentas específicas como Scratch, Lego Robôs, Raspberry Pi, CS Unplugged e App Inventor para o ensino e incorporação do pensamento computacional e de ciência da computação em suas escolas. Os 24 professores que foram inquiridos sinalizaram que a ferramenta mais intuitiva e fácil de aprender é o Scratch.

3.1.2.
App Inventor

App Inventor é uma linguagem de programação visual projetada pelo Google, que o lançou em julho de 2010. Nesse momento, o App Inventor era um serviço web gratuito que foi fornecido ao público como parte da suíte do Google Labs. Em agosto de 2011, o Google anunciou que o App Inventor seria lançado como um projeto de código aberto. O Centro de Aprendizagem Móvel foi estabelecido no MIT Media Lab para continuar fornecendo o App Inventor ao público. No quarto trimestre de 2011, o Center of Mobile Learning iniciou 3 trabalhos no App Inventor e em março de 2012, uma versão beta do MIT App Inventor foi lançada ao público.

MIT App Inventor é uma ferramenta de introdução à programação e à criação de aplicações que transforma o complexo código textual em código visual, isto é, usando a técnica drag and drop para construção do script de código. Tem uma interface gráfica simples, que permite aos iniciantes criar suas próprias aplicações em questão de horas com só algumas aulas.

No estudo feito por Fronza et al. (2015), eles descrevem a estrutura de um curso que foi chamado MobileDev e que foi realizado no verão na escola. Através do curso eles usam a curiosidade dos estudantes para desenvolver aplicações mobile e introduzir com essas aplicações os conceitos de pensamento computacional. Eles usaram o App Inventor como framework para suas aulas.

Grover & Pea (2016) usaram o App Inventor para pilotar o curso que eles desenharam sobre "Computational Discourse". Os alunos eram introduzidos nas ideias de ciência da computação através do desenvolvimento de competências no pensamento computacional. O curso baseia-se numa estrutura que enfatiza a importância da interação social no desenvolvimento de processos individuais. O estudo piloto foi um curso de um dia com sete estudantes do ensino médio com idade média de 13 anos. O curso foi dividido em duas sessões distintas.

A primeira foi para apresentar os conceitos básicos do App Inventor. Na segunda sessão, os alunos trabalharam em pares ou sozinhos para desenvolver um aplicativo de sua escolha. Os resultados preliminares mostraram que o vocabulário de ciência da computação aumentou e observou-se um crescimento no uso de importantes elementos do pensamento computacional no contexto do

desenvolvimento dos aplicativos. Eles sentiram que o App Inventor é uma boa ferramenta para este tipo de ensino, pois conduziu a discussões sobre como resolver problemas, e sua arquitetura baseada em eventos permitiu que os alunos falassem sobre eles de forma como os programadores novatos abordam a solução de problemas. Eles também observaram que o App Inventor é bem parecido com o Scratch e Alice, mas oferece os benefícios de ter algo muito tangível para mostrar e permitir que os alunos façam aplicativos que lhes interessem e que tenham muita interação.

3.1.3. Lego Mindstorms

A série de kits Lego Mindstorms contém software e hardware para criar robôs customizáveis e programáveis. Eles incluem o seguinte: um minicomputador, um conjunto de sensores e motores modulares e peças de Lego para criar os sistemas mecânicos. Os kits Mindstorms são vendidos comercialmente e são usados como ferramenta educacional.

Van Dyne & Braun (2014) desenvolveram um curso em Montana Tech (Universidade de Montana) que teve como alunos estudantes de ciências da computação e engenharia com fortes habilidades matemáticas. Eles dividiram a matéria em duas partes: teoria e prática. Na parte teórica eles usaram apostila de robótica e na prática usaram o kit de robô Lego Mindstorm NXT. Eles simularam os conhecimentos adquiridos na parte teórica. Originalmente usaram a linguagem visual de programação próprio do lego, depois usaram o Lejos, um ambiente de programação textual que suporta a programação em Java. Entre os tópicos que foram ensinados estão algoritmos, dados e variáveis, iteração e ordenamento. Eles descobriram que o curso incrementou as habilidades de análise dos estudantes. Para isso, usaram o método de avaliação de Whimbey Analytical Skills Inventory (Whimbey, A. et al., 2013), estes testes foram realizados antes e depois do curso.

Bers, M. et al. (2010, 2014) desenvolveram um curso chamado "*TangibleK*", para estudantes de ensino básico. No curso eles ensinaram tópicos do pensamento computacional através da programação de robôs. Eles usaram o CHERP (Creative Hybrid Environment for Robotics Programming), uma linguagem para a programação de kits robóticos. CHERP é uma linguagem visual de programação desenhada para estudantes novatos em programação. Os tópicos

que foram ensinados são sequências, loops e funções. A aplicação mais destacada foi fazer que o robô dance ao ritmo do baile "*Hokey-Pokey*" e usaram os sensores para ligar as luzes quando o robô está no escuro. Eles mostram com seus resultados que "*TangibleK*" serviu para motivar os estudantes a aprender programação e se interessar mais pela tecnologia.

3.1.4. CS Unplugged

CS Unplugged é uma coleção gratuita de atividades de aprendizado que ensinam Ciência da Computação através de jogos e quebra-cabeças que usam cartas, cordas, lápis de cera, entre outras coisas. Foi desenvolvido pelo Education Research Group na Universidade de Canterbury, Nova Zelândia, para que os estudantes dos primeiros anos possam interagir com ciência da computação, resolvendo as diferentes questões e desafios que os pesquisadores elaboram, isto sem ter que aprender nenhuma linguagem de programação antes.

No estudo feito por Dwyer et al. (2014) discute-se a necessidade de desenvolver habilidades do pensamento computacional em todo o currículo, enfocando-se especificamente em reforçar o fenômeno da física, desenhando jogos em Scratch usando os conceitos físicos como gravidade, aceleração, etc. A principal pesquisa discutida, foi feita em uma atividade CS Unplugged chamada "*Marching Orders*", que consiste em que um aluno dá a outro aluno um conjunto de instruções sobre como desenhar uma figura geométrica. Durante as reuniões em grupos eles descobriram que pequenos erros nas instruções podem mudar toda a figura geométrica que eles fizeram.

Pollock et. al. (2015) desenvolveram um curso para o treinamento de alunos em ciências da computação e em STEM em seus escolas. Usaram Scratch como primeira ferramenta, depois usaram as aplicações do site "CS Unplugged" para ensinar tópicos mais avançados. Os pesquisadores verificaram que os alunos que seguem esta sequência têm mais familiaridade com os conceitos de ciências da computação.

3.1.5. AgentSheets e AgentCubes

AgentSheets e AgentCubes são ferramenta que foram realizadas pela equipe Scalable Game Design (SGD) no Departamento de Ciência da Computação,

Universidade de Colorado Boulder, EUA. Estas ferramentas permitem que as pessoas criem seus próprios jogos e animações e publiquem-nas na Web através de uma interface *Drag and Drop* fácil de usar. As animações interativas ajudam os alunos a compreender novas ideias, testar teorias, explorar processos complexos em vários campos científicos. Eles afirmam que a construção de jogos ensina conceitos de computação informática, lógica e pensamento algorítmico (Agentsheets, 2017).

Em um estudo feito por Basawapatna et al. (2014) eles investigaram como o grupo SGD integra a simulação e modelagem em tarefas e dá diferentes estratégias para realizá-lo. Eles mostram o enlace entre simulação e CT. Durante a animação os usuários "começam com uma pergunta, desenvolvem um modelo, expressam o modelo computacionalmente, executam o modelo, visualizam as consequências de seus pensamentos e revisam o modelo". Os pesquisadores deram uma introdução para que os alunos criem suas próprias simulações. Esta introdução é constituída por sete estágios que são os seguintes:

- Animações - assistindo um filme ou similar
- Animações interativas - uma animação que o usuário pode alterar certos parâmetros
- Animações coletivas - animações que tenham algum elemento social
- Programação do usuário final - usando ferramentas como AgentCubes ou Scratch
- Programação tradicional - usando linguagens como Java ou C ++

Para cada uma das etapas, eles fornecem ferramentas e exemplos de como ele pode ser feito.

3.2. Ferramentas em 3D

Estas ferramentas oferecem uma interface de saída 3D. Elas são ferramentas que usam imagens 2D para a montagem de blocos para desenhar o código fonte, mas a execução do código é feita em três dimensões, podendo-se gerar animações, histórias ou videogames. As ferramentas mais conhecidas são apresentadas a seguir.

3.2.1. Alice

Alice é uma ferramenta para criação e animação de mundos tridimensionais, definida para ser fácil de programar e modificar. A programação em Alice é feita seguindo o paradigma Orientado a Objetos (POO). O desenvolvimento do software iniciou-se na "University of Virginia" e posteriormente continuou com uma equipe na "Carnegie Mellon University". É uma ferramenta que pode ser utilizada por pessoas que não sabem programação, evitando as etapas demoradas de aprendizado para utilização de tecnologias dessa natureza (Pierce et al., 1997). Alice foi desenvolvida inicialmente em Python. Por ser interpretada, Python propicia facilidades na alteração e validação de código, sendo que mudanças efetuadas podem ser observadas em segundos, e por esta razão é mais rápida que C ou C++ para prover alterações.

De forma a evitar que usuários necessitam estudar e aprender conceitos relacionados a computação gráfica ou realidade virtual, como transformações (escala, rotação, translação, matrizes, coordenadas homogêneas, etc.), Alice provê suporte a requisitos de computação gráfica e realidade virtual, sem a exigência do aprendizado destes conceitos. Alice pretende evitar que o desconhecimento destes conceitos seja fator desmotivador.

Ademais, usuários podem ser levados a cometer erros de sobreposição de transformações, como uma aplicação de transformação de escala, que pode levar à modificação da figura visualizada, de uma forma não intuitiva.

No estudo feito por Daily, S. B et al. (2014) os pesquisadores falam sobre a importância de criar aplicações onde os participantes se sintam representados pelos atores da aplicação, isto é, que o participante se sinta imerso dentro da aplicação. Para isso, eles desenvolveram uma aplicação usando a ferramenta de Alice onde os estudantes dão movimentos de dança a um boneco. Os estudantes podem gerar um script de código na qual dão os seguintes movimentos ao boneco: movimento das mãos, movimentos das pernas e movimento da cabeça. Os pesquisadores tiveram como resultado que os estudantes se sentiram motivados pela tecnologia e gostaram de gerar todas as combinações de movimento. Com isto, os pesquisadores ensinaram aos estudantes conceitos do pensamento computacional como: algoritmos, funções, abstração, paralelismo e debugs.

3.2.2.
Ferramentas de Realidade Virtual – VenVi

Esta aplicação foi construída pelo grupo de pesquisa da universidade CLEMSON. "*A VEnvI (Virtual Environment Interactions) é uma aplicação na qual os alunos aprendem conceitos de ciência da computação através do processo de coreografia de movimento para um personagem virtual usando uma interface divertida e intuitiva"* (Parmar et al., 2016).

Um estudo feito por Parmar et al (2016), na qual participaram 54 estudantes tive como objeto a avaliação desta ferramenta, e na qual eles tinham que criar uma performance de uma dança para um avatar virtual, isto montando os blocos que e formando um script de código, destes 54 participantes só 16 participaram em uma experiência imersiva virtual, onde o participantes era representado por um avatar (masculino ou feminino) e onde eles observavam o boneco dançar em função do script de código feito. Os pesquisadores encontraram que os participantes gostaram da ferramenta e motivou a eles a aprender mais sobre os conceitos de programação e pensamento computacional, além disso durante a experiência de imersão eles mediram a Tele presença, Presença Social, e Usabilidade, próprias de uma ferramenta Virtual.

Uns grandes números de ferramentas foram desenvolvidos para ensinar o pensamento computacional; estas ferramentas vão desde jogos até aplicações científicas. Podemos notar o esforço dos pesquisadores em fazer com que as ferramentas sejam divertidas e que cheguem a todos os estudantes de todas as idades, gêneros e habilidades. Os professores de diferentes matérias também se beneficiam, já que estas ferramentas podem ser incluídas nas aulas que eles ensinam.

É por isso que para desenvolver a ferramenta do presente trabalho utilizou-se algumas funcionalidades das ferramentas anteriormente mencionadas como a técnica de *drag and drop* para a montagem dos blocos e ofereceu-se um conjunto de desafios cuja execução do código fonte feita pela montagem dos blocos vai ser vista em um ambiente de Realidade Virtual, e também se tomou em conta o desenho colorido da interface gráfica.

3.3.
O uso de pensamento computacional em outras matérias.

Para muitas escolas, em qualquer parte do mundo, é difícil executar um currículo completo de ensino de ciência da computação além das matérias comuns que as crianças tradicionalmente assistem. Uma das vantagens da ciência da computação é que ela pode ser ensinada de uma forma indireta, isto é, inserindo alguns tópicos nas matérias comuns.

Nesta seção resume-se alguns estudos realizados de como o pensamento computacional foi inserido nos outros cursos que os alunos assistem em suas escolas. Além disso, sobre como ensinar aos professores das escolas como eles poderiam inserir o pensamento computacional em suas aulas e com que ferramentas começar (Lockwood, J. et al. 2017).

3.3.1.
Ciências

No estudo feito por Ahamed et al. (2010), onde se desenvolveu um workshop, eles enfatizaram o seguinte *"O que têm em comum as ciências naturais, matemáticas e ciências da computação? "*, o artigo mostra os tópicos mais relevantes do qual eles conversaram.

- Pensamento computacional - uma introdução na qual eles explicaram tópicos como abstração, algoritmos, etc.
- Simulações - introduziu professores aos conceitos de simulações e sua relevância, eles simularam as leis de Newton
- Probabilidade - usaram o VPython para construir uma simulação, além de discutir a relevância da probabilidade em diferentes áreas
- Matemática - criptografía usando a ferramenta "*CS Unplugged*"
- Física – Ensinou-se plano inclinado, segunda lei de Newton, gravidade, etc.
- Empregos de Ciência da Computação - trazer consciência de quais empregos estão lá fora.

Os autores mencionam que no estudo realizado eles verificaram que alguns estudantes já têm conhecimento da ciência da computação e isto permite aos investigadores seguir melhorando e aplicar novas ferramentas.

No estudo feito por Hambrusch et al. (2009) se descreve o desenvolvimento do curso *"Introdução ao pensamento computacional*" que foi dado pelos professores de ciências. O curso tinha como principal objetivo ensinar o pensamento computacional valendo-se de outras áreas como a física, química e a bioinformática. A ferramenta que usaram para o ensino da matéria foi Python e o modelo de Monte Carlo. Em física ensinaram a simulação do gás ideal, em bioinformática ensinaram grafos para modelar as proteínas. Além disso, os professores falaram sobre a história da ciência da computação assim como os limites e o futuro dela.

Para afiançar os conhecimentos dos alunos os professores elaboram quatro projetos:

- Manipulação de áudio digital
- Experimentos computacionais sobre percolação em grade
- Simulação de sistemas físicos
- Análise das interações proteína-proteína

Os pesquisadores verificaram que o curso incrementou o interesse dos alunos, por isso eles abriram outro curso de reforço, o qual teve uma demanda muito grande por parte deles. Desta experiência eles sugerem que Python é a melhor ferramenta para começar com o ensino da programação.

3.3.2. Biologia

No estudo feito por Rubinstein & Chor (2014), eles discutem principalmente como eles ensinam conceitos de algoritmos, abstração e pensamento lógico aos estudantes de biologia. Este curso requer um conhecimento mínimo de programação e está focado em "*Desenvolver as habilidades do pensamento computacional nos estudantes*". O curso tem quatro módulos, em cada módulo se ensina um tópico diferente da matéria da biologia. Se fez um mapeamento dos tópicos de biologia com os tópicos de ciências da computação e conseguiram o seguinte:

- Redes biológicas → Grafos, algoritmo de Dijkstra → algoritmos gulosos, abstração, redução
- Sequências biológicas → Autômatos e expressões regulares → Pré-processamento

- Simulação de sistemas → Redes booleanas/discretas → Simulação, matemática discreta
- Imagens biológicas → Processamento de imagens: detecção de arestas → Desenho modular

Os estudantes resolvem problemas da vida real usando a linguagem de programação Python e eles se concentram no uso prático, mais que no erro de sintaxes. Eles falam sobre as vantagens de conhecer a programação e declaram ter um certo "*incômodo*", em porquê não terem feito a matéria de programação anteriormente.

3.3.3. Matemática

No estudo feito por Sysło & Kwiatkowska (2014), os autores discutem como o pensamento computacional pode ser incorporado em uma matéria tradicional de matemática e como ele pode ajudar na melhoria da aprendizagem da matéria. Eles trabalham com os livros da escola de primeiro nível; estes livros têm conteúdos relacionados à informática. Os exemplos que os pesquisadores consideram fazer são:

- Representação de números → polinômios
- Redução e composição → Dado lados de um triângulo, conhecer se é um triângulo válido
- Aproximação → erros de arredondamento → equação quadrática
- Recursão → difícil para matemática, mais fácil depois do ensino em ciências da computação
- Pensamentos heurísticos → algoritmos de prova e erros, gulosos, caminho mais curto, etc.

No estudo feito por Freudenthal et al. (2010), é apresentado o conteúdo e a avaliação de um curso introdutório de programação chamado "*Pensamento computacional com suporte de aplicações*" (MPCT). Este curso está desenhado para a aprendizagem de tópicos básicos de pré-cálculo para estudantes de primeiro ano. O curso está feito com uma série de módulos de cálculo onde o aluno programa conceitos de álgebra e geometria.

3.3.4. Multidisciplinar

No estudo feito por Goldberg et al. (2012) é apresentado um curso no qual eles principalmente aplicam a programação nas seguintes matérias: arte, biologia e matemática. Eles incluem atividades baseadas em ciências da computação que permitem aos estudantes raciocinar e conhecer de maneira indireta os conceitos da programação. Eles posteriormente discutem a importância da programação e buscam de que forma melhorar o conteúdo das matérias. Eles dão alguns exemplos de como foi incluído em cada uma das matérias.

- Arte – usam Photoshop, Dreamweaver e outras ferramentas para aprender sobre gráficos vetoriais, usam Arduino e outras ferramentas para ensinar programação, sensores etc.
- Biologia – Tipos de DNA → algoritmos, análise de dados
- Educação em saúde - ensinou sobre "*querys*" para a busca de informação adequada.
- Geografia - cartografia computacional, visão computacional
- Sociedade civil e social - fez rastreadores usando o Arduino para conhecer as ruas das cidades.

Eles acham que a ciência da computação é fácil de ser aplicada nas outras matérias e que em um futuro todas as outras escolas poderão inserir o mesmo plano curricular.

No estudo feito por Shailaja & Sridaran (2015), eles dão uma visão geral de porquê devemos ensinar o pensamento computacional. Eles dão uma lista de tópicos que são baseados no modelo ACM/IEEE. Este modelo inclui: algoritmos, estruturas discretas, fundamentos de programação, visão computacional, entre outros. Eles propõem diferentes tópicos de acordo a idade do estudante, suas propostas são:

- 2do ano do Ensino Fundamental (Montessori) - se integram com tópicos já existentes
    - Visualização - Multimídia como um CD educacional
    - Reconhecimento de padrões – Buscam que o aluno trabalhe de forma independente no software, como o Paint do Microsoft Windows

- Idade (3-5)
    - Decomposição - usa LOGO/Scratch para aprender sobre ângulos, execução de comando etc.
    - Pensando em Algoritmos – Escrita de algoritmos
    - Avaliação - aprendizagem independente; usam desafios como: puzzles e jogos plataforma de autoaprendizagem
- Idade (6-8)
    - Abstração - mudanças na tecnologia
    - Pensamento Crítico - solução de problemas, o aluno propor uma solução
    - Computação - usa ferramentas de software (Visual Basic) para mostrar o que é a programação, escrevendo programas simples
- Idade (9-12)
    - Pensamento computacional – Os alunos já podem aprender linguagens de programação como Python ou Java. Eles devem desenvolver jogos que utilizem operações matemáticas.

Os trabalhos de pesquisa que foram expostos indicam que o ensino do pensamento computacional não exige que se criem novas matérias. Nota-se que o pensamento computacional foi inserido em diferentes disciplinas comuns, isto nos mostra que depende do professor a iniciativa de inserir a programação dentro de suas matérias.

É por isso que o presente trabalho leva em consideração o ensino do pensamento computacional, dando para o estudante uma introdução aos elementos que a compõem, tais como conceitos (lógica, algoritmos, decomposição e avaliação) e comportamentos (desenho da solução e a depuração). Com o presente trabalho se tem a base necessária para que em um futuro se possa integrar algumas animações e videogames que tenham o conteúdo de outras matérias como Biologia, Anatomia, Geografia entre outros.

4
Metodologia de desenvolvimento

O processo para o desenvolvimento da ferramenta é adaptado do RUP (Rational Unified Process) (Robertson and Robertson, 2006) e a metodologia para o desenvolvimento de sistemas virtuais de Sherman and Craig, (2003). Na Figura 4.1 mostram-se as 4 fases de desenvolvimento: Análise e Requisitos, Processo de Desenho, Desenvolvimento do Sistema e Processo de Avaliação.

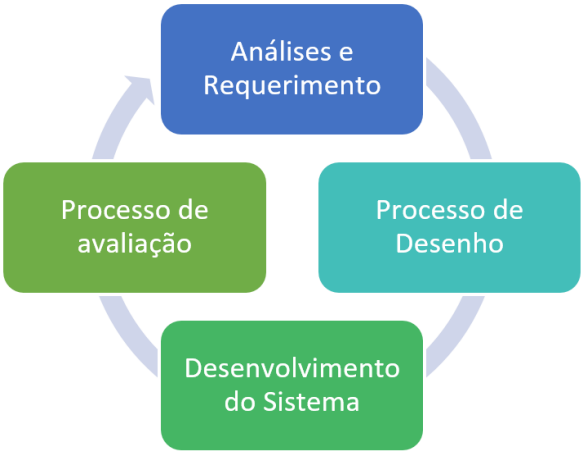


Figura 4.1 Fases adaptadas do RUP para o desenvolvimento do software (Robertson et al, 2006)

A fase de Análise e Requisitos consiste em identificar os Stakeholders e definir de forma geral como vão participar no sistema. *"Um requisito é algo que o produto deveria fazer ou a qualidade que o produto deveria ter"* (Robertson e Robertson, 2006). Os requisitos podem ser funcionais ou não funcionais. Os requisitos funcionais são aqueles que o produto deveria fazer para ser útil no domínio do cliente e os requisitos não funcionais são propriedades ou qualidades que o produto deve ter para que este seja melhor.

A fase de Processo de Desenho consiste em desenhar as planilhas e desenho de classes que darão suporte ao sistema, nela se usaram o Sofware StarUML para o modelado de artefatos. Entre os artefatos utilizados para o desenvolvimento do

sistema temos: diagrama de Caso de Uso do Sistema, Arquitetura do Sistema y Diagrama de blocos.

A fase de Desenvolvimento do Sistema consiste em definir a gramatica que deverá cumprir a linguagem de programação, para isso, usara-se uma gramatica de disposição de imagens (símbolo e significado) o *símbolo* representa a imagem do bloco e o *significado* uma determinada estrutura de código fonte.

A fase de Processo de Avaliação consiste em verificar se a ferramenta desenvolvida proporcionava um ambiente no qual os estudantes se sentissem motivados por aprender mais sobre a programação. e avaliando se a sensação de imersão ajuda para que os estudantes estejam motivados por cumprir todos os desafios que a ferramenta propõe.

Para nossa ferramenta de Realidade Virtual, primeiro definiremos quem são nossos usuários principais ou stakeholders.

Identificando aos Stakeholders e Cenários

O sistema Virtual tem um StakeHolder principal: o estudante.

- **O Estudante:** É a pessoa que se submerge no sistema virtual e que interage com os dispositivos de entrada e saída (Microsoft Kinect, Capacete Oculus Rift). Na Figura 4.2 mostra-se o estudante sentado em uma cadeira em frente ao sensor do Microsoft Kinect e capacete Oculus Rift.

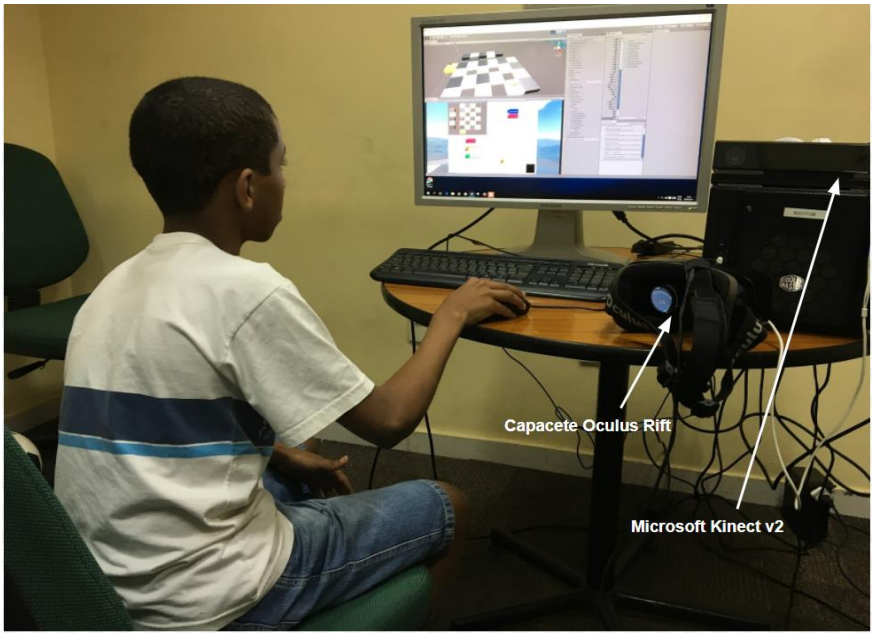


Figura 4.2 Usuário navegante

- **Descrevendo o Cenário do Ambiente Virtual:** O cenário onde se desenvolverá a simulação consiste em um oceano. Dentro do oceano tem uma superfície que forma um labirinto por onde o carro terá que percorrer. O oceano tem a característica que suas águas se movimentam formando pequenas ondas; este movimento pretende dar sensação de realismo para o usuário. Na Figura 4.3 mostra-se a cena principal que contém o labirinto e ao final tem o conjunto de pessoas que formam a audiência. Além disso pode-se ver os braços direito e esquerdo do objeto humanoide, que representa o participante dentro da animação.

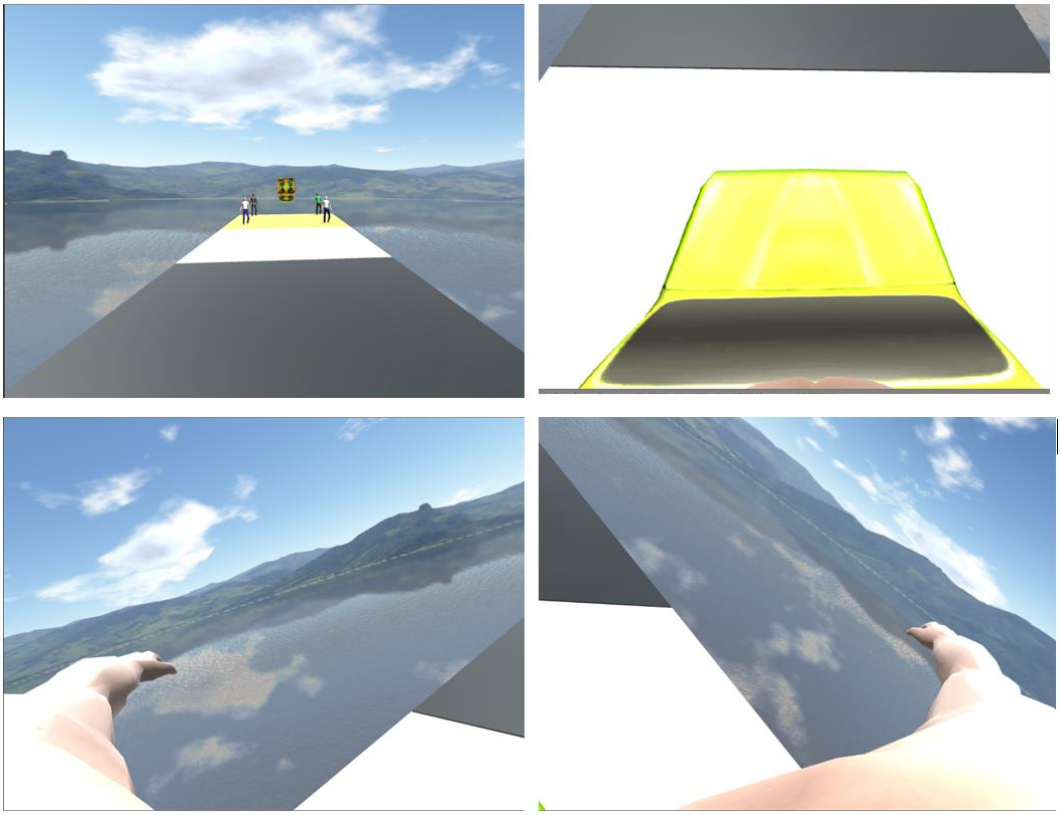


Figura 4.3 Cenário principal, pode-se ver o carro acima da superfície que forma o labirinto dentro do Oceano (acima). Braços do avatar representando o usuário (abaixo).

- **Descrevendo o Cenário dos Bloques.**

A ferramenta de Realidade Virtual tem um componente que serve para a elaboração do código fonte, é neste componente onde se encontram os blocos que representam as estruturas de controlo condicional e repetitivas com as quais se desenvolveram o Script de código que permita solucionar os 10 desafios. Na

figura 4.4 pode-se ver a janela onde o script de código será gerado. Nela aparecem numeradas as funcionalidades (1) Opções de edição de código “Executar código, Mostrar código, Limpar”, (2) Opções de configuração para um determinado usuário “Seletor de nível, Seletor de gênero, Aplicar”, (3) Área para a elaboração do código com o montagem de blocos, (4) Conjunto de blocos “Blocos de Controlo, Blocos de Funções”

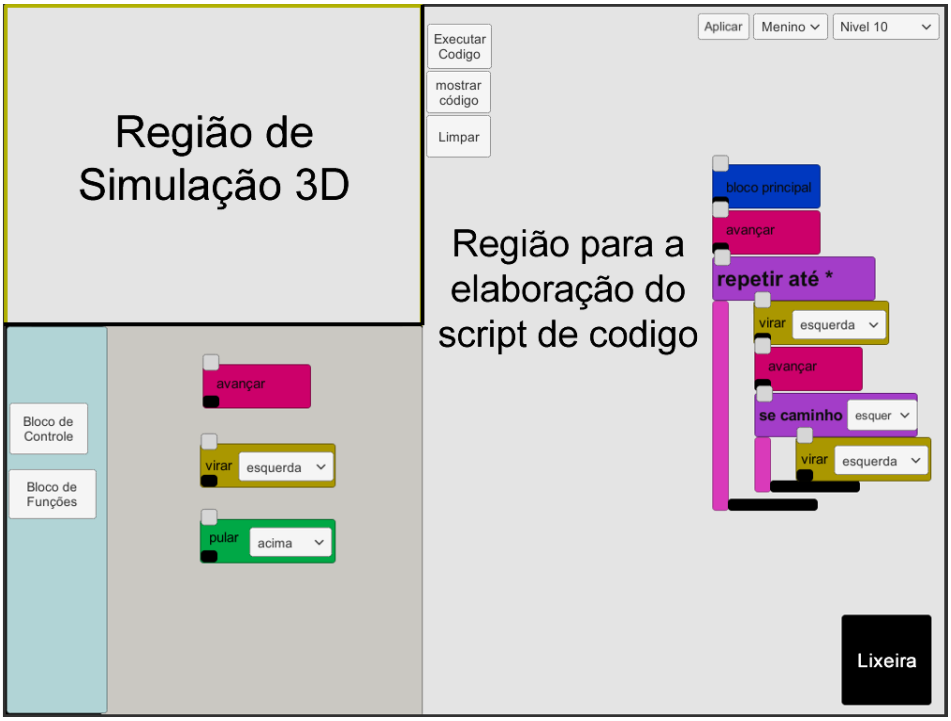


Figura 4.4 Figura 4.4: Região onde o Script de código fonte será gerado, pode-se ver (1) Opções de edição de código, (2) Opções de configuração para um determinado usuário, (3) Área para a elaboração do código com a montagem de blocos, (4) Conjunto de blocos.

4.1. Análise e requisitos da ferramenta

A seguir mostra-se a lista de requisitos funcionais e não funcionais que serão necessários para a construção do sistema.

4.1.1. Requisitos Funcionais da ferramenta

Para a especificação dos requisitos funcionais foram abordados três enfoques possíveis, permitindo com eles uma melhor descrição do sistema:

- **Enfoque 1.** Quando o ambiente virtual reproduz tarefas e ações executadas por usuários em um ambiente real e se pretende retratar esta situação como ela é, ou seja, com realismo.
- **Enfoque 2.** Quando os usuários desempenham determinadas tarefas no mundo real, mas se decide que o ambiente virtual reproduz tais tarefas de forma diferente ao que ocorre na realidade.
- **Enfoque 3.** Quando o ambiente virtual pretendido compreenderá a realização de tarefas e interações que não são desempenhadas no mundo real.

Tendo estes 3 enfoques, a Tabela 4.1 mostra os requisitos necessários para o desenvolvimento do sistema, estes requisitos funcionais foram obtidos das diferentes ferramentas de programação visual em especial (Scratch, Alice, AppInventor e Venvi). Além disso tomou-se em consideração a experiência do pesquisador, isto por haver trabalhado muito tempo no ensino da programação a estudantes de ensino fundamental.

Tabela 4.1 Requisitos Funcionais

Cod	Funcionalidade	Descrição
RF01	O Sistema deve permitir a criação de um documento novo	O Sistema apresentará a opção "criar documento novo", esta opção apresentará uma janela branca contendo só o bloco inicial sem nenhum outro bloco (para isso o sistema deverá apagar todos os blocos que estavam na janela). A nova janela apresentará as seguintes opções: executar código, mostrar código, mudar de nível, limpar janela, sair da aplicação, e criar documento novo.
RF02	O Sistema deve permitir que usuário manipule os blocos de programação	O sistema permitirá que o usuário possa levar um bloco da lista de blocos faz a janela de código, onde se formará o script de código.
RF03	O sistema deve permitir a concatena. dos blocos	O sistema poderá unir dois blocos, para isto a ferramenta os juntará automaticamente sempre que os blocos estejam sobrepostos.

RF04	O Sistema deve permitir encapsular blocos dentro dos blocos compostos	O sistema permitirá que os blocos sejam postos dentro dos blocos compostos, isto só quando os blocos estejam sobrepostos acima do bloco composto.
RF05	O sistema deve permitir aumentar o tamanho dos blocos	O Sistema deve permitir que os blocos compostos estejam na capacidade de aumentar de tamanho vertical para encapsular aos outros blocos.
RF06	O sistema deve permitir diminuir o tamanho dos blocos	O Sistema deve permitir que os blocos compostos estejam na capacidade de reduzir seu tamanho vertical, isto quando os blocos de seu interior sejam retirados ou removidos.
RF07	O sistema deve permitir a execução dos algoritmos desenvolvidos	O Sistema deverá apresentar para o usuário uma opção de execução, e depois de que esta opção seja selecionada a aplicação começará a funcionar.
RF08	O Sistema deve permitir apresentar o código fonte gerado pelo usuário.	O sistema deverá apresentar a opção “mostrar código fonte”, o sistema deverá apresentar o código fonte textual na linguagem C# em uma janela, esta janela terá a opção fechar janela.

Da lista de requisitos funcionais se obteve a seguinte lista de casos de uso:

- Iniciar sessão de treinamento
- Criar novo documento
- Mexer bloco
- Unir/Separar automaticamente os blocos
- Aumentar/Diminuir automaticamente o tamanho vertical dos Blocos
- Encapsular/Descapsular os blocos
- Executar a aplicação
- Mostrar código fonte

Estes casos de usos serão considerados para o desenvolvimento da ferramenta.

4.1.2.
Requisitos não funcionais.

Para o desenvolvimento da Ferramenta Virtual considerou-se os fatores e condições externas que devem ser cumpridas; para isso a Tabela 4.2 mostra a lista de Requisitos não funcionais que devem ser levados em consideração, estes requerimentos formar obtidos da experiência do pesquisador.

Tabela 4.2 Lista de Requisitos não funcionais

Cód.	Requisito não funcional	Descrição
RNF01	Desempenho	O sistema deve usar algoritmos que consomem poucos recursos computacionais como: 8 Gb de memória RAM, 2Gb de memória dedicada e 2.7 Ghz de processador.
RNF02	Usabilidade	O sistema deve ser intuitivo e fácil de usar, com fácil navegação por suas cenas (oceano e labirinto), também durante a manipulação dos blocos, isto é, quando o usuário gera um script de código mediante a montagem de blocos.
RNF03	Segurança	O sistema permitirá que só um usuário navegue dentro do sistema. O sistema está desenvolvido em um ambiente local não sendo necessário o uso de internet.
RNF04	Portabilidade	O sistema virtual será executado no S.O. Windows 8 de 64 bits. Será implementado usando a ferramenta de programação Unity 5. O sistema poderá ser executado nos seguintes sistemas operacionais (Windows 7 SP1+, 8, 10; Mac OS X 10.8+.)
RNF05	Confiabilidade	O sistema deve ser desenvolvido respeitando todas as especificações de Caso de Uso de Sistema, modelo de arquitetura e os outros artefatos de modelagem.
RNF06	Tridimensional.	O ambiente virtual será desenvolvido com a

		ferramenta Unity que permite desenvolver objetos e ambientes 3D e permite percorrer o ambiente mediante câmeras.
RNF07	Multissensorial.	O sistema virtual deve possuir a capacidade de estimular os sentidos do usuário: para a visão temos o capacete (Óculos Rift), para a telepresença (ver seu próprio corpo no cenário 3D) temos o sensor de Microsoft Kinect.
RNF08	Interativo	O sistema deve permitir detectar o evento que o usuário realiza, como por exemplo, o usuário mexer os braços e pernas.
RNF09	Realístico	O sistema deve ser construído com realismo que permita ao usuário pensar que está em um ambiente real.

4.2. Projeto da Ferramenta

4.2.1. Diagrama de Casos de Uso do Sistema

Dos requisitos funcionais acima descritos, obtemos o diagrama de Caso de Uso do Sistema na Figura 4.5; este diagrama foi feito com a ferramenta de desenho StarUML (StarUML, 2017).

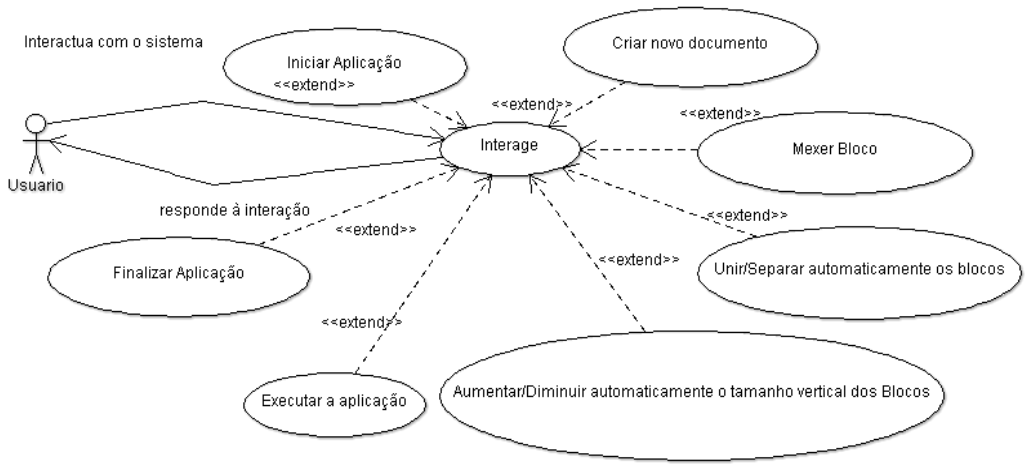


Figura 4.5 Diagrama de caso de uso do Sistema

A especificação de cada Caso de Uso se encontra no Apêndice A.

4.2.2.
Arquitetura do Sistema

A arquitetura da ferramenta possui três componentes: *Biblioteca de elementos*, *Motor de cenário e Cena do usuário.* Motor de cenário é o componente principal, pois permite combinar as tarefas que o usuário quer fazer mediante os Inputs/Outputs dos dispositivos com o cenário virtual. O componente *Biblioteca de elementos* está associada com outras aplicações como o banco de dados e bibliotecas externas. O componente *Cena do usuário* mostra os ambientes virtuais dependendo das funcionalidades (input do dispositivo e lógica de programação) que o usuário usa. Na Figura 4.6 mostra-se a arquitetura da ferramenta.

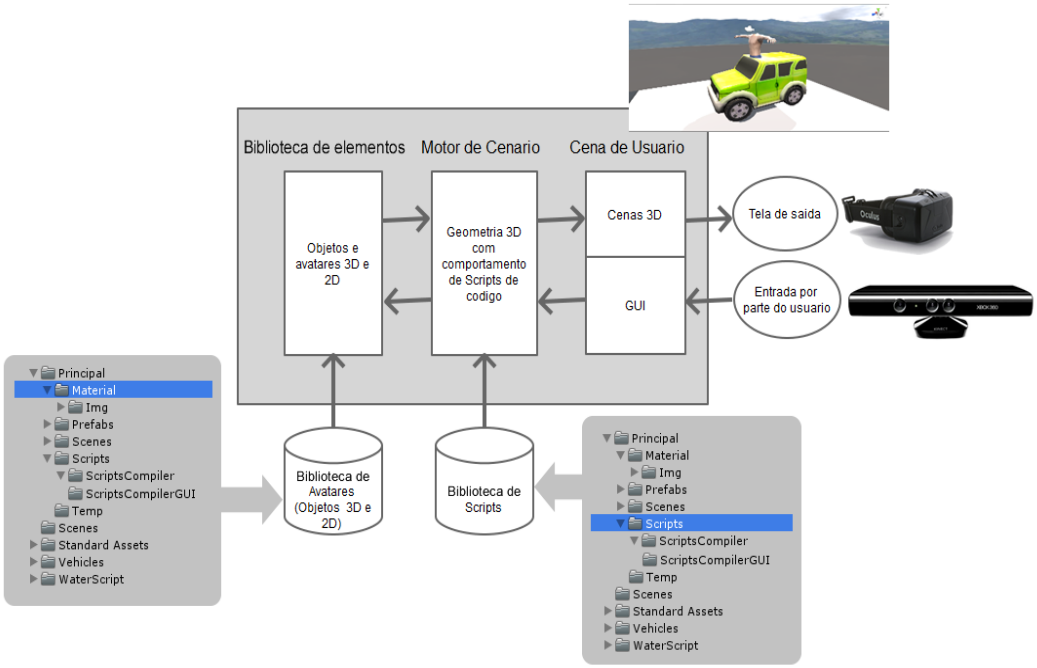


Figura 4.6 Arquitetura da ferramenta

A seguir descreveremos os componentes desta arquitetura.

- **Biblioteca de Elementos:** Este componente funciona como um repositório de objetos 3D, onde estarão armazenados os diferentes objetos que serão necessários para o desenvolvimento da ferramenta; estes objetos são: **carro 3D, cofre do tesouro**, estes foram obtidos da própria biblioteca de avatares do site da Unity; **lagoa**, foi elaborado tendo em conta o movimento das ondas da água; **o labirinto,** este objeto foi desenhado por um conjunto de objetos

cubo 3D, que são parte da biblioteca da Unity; **Avatar humano e a audiência**, foram desenhados 3 modelos de avatar humano, um para participantes homem, outro para as mulheres e para a audiência; as **Montanhas e o céu,** foram obtidos da própria biblioteca de objetos do Unity.

- **Motor de Cenário:** O motor de cenário relaciona os comportamentos, cenas, diferentes scripts e os objetos 3D no espaço. **Biblioteca de Scripts:** Este componente atua como um repositório de todos os Scripts que foram desenvolvidos para que a ferramenta funcione, nela podemos ver duas subpastas as quais são: **ScriptCompiler** e **ScriptCompilerGUI**. A primeira pasta armazena os Scripts de código fonte para a execução da animação; esta animação vai depender do algoritmo que o usuário desenvolve. A segunda pasta armazena os Scripts da Interface Gráfica de Usuário (GUI). Esta GUI apresenta para o usuário o conjunto de blocos e permite que o usuário realize todas as funcionalidades que foram descritas como requisitos.
- **Cena de Usuário:** É a interface de comunicação entre o usuário e a ferramenta, a fim de realizar suas tarefas e alcançar seus objetivos.

4.2.3. Diagrama de blocos da ferramenta

Na Figura 4.7 podemos ver os blocos que são usados na construção da ferramenta.

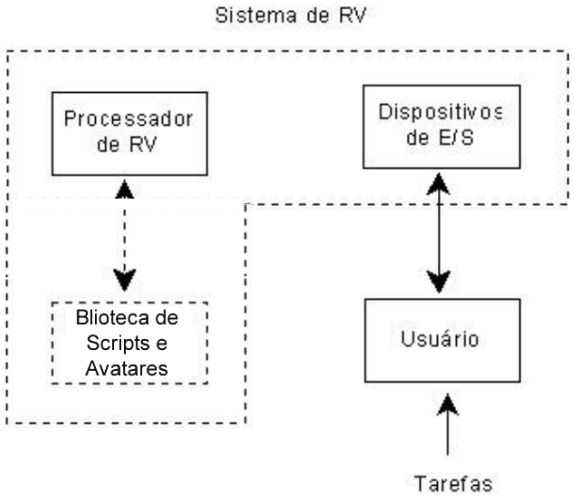


Figura 4.7 Diagrama de Blocos da ferramenta

- **Processador de Realidade Virtual:** Este bloco apresenta o software que está formado por todos os algoritmos e bibliotecas próprias do Unity usados para a execução da ferramenta.
- **Dispositivo de E/S:** Este bloco apresenta os dispositivos de entrada e saída da ferramenta. *Os dispositivos de entrada* são aqueles que provêm informação ao Sistema, por exemplo, temos o Microsoft Kinect que manda informação de posicionamento dos braços, pés e cabeça do usuário. *Os dispositivos de saída* são aqueles que recebem informação do sistema de treinamento para depois exibi-lo, por exemplo, a janela do computador. *Os dispositivos de entrada e saída* são aqueles que ingressam informação ao sistema como também são os que recebem informação dele, por exemplo, temos o capacete "Oculus Rift", que manda a informação de giroscópio ao sistema; se o usuário gira a cabeça para a direita o sistema mostra o ambiente do lado direito e se gira a cabeça para esquerda o capacete gira para a esquerda.
- **Biblioteca de Scripts e Avatares**: Este bloco apresenta todo o código fonte e Avatares que foram feitos pelo desenvolvedor para a construção da ferramenta.
- **Usuário:** Este bloco representa a pessoa que vai ser imersa no ambiente virtual e vai ser quem vai usar os dispositivos de entrada e saída. No nosso sistema de realidade virtual, vai ser o estudante.

4.3. Desenho da experiência do usuário.

Nesta seção se descreve a parte gráfica de interface do usuário e os componentes externos que são parte do sistema completo.

O desenho da ferramenta foi inspirado pelas tecnologias já existentes que têm como finalidade o ensino da programação mediante a construção dos blocos. As ferramentas que foram selecionadas como modelo foram: Scratch, Alice e App Inventor. Estas ferramentas têm em comum o desenho dos blocos, assim como as funcionalidades do compilador visual. Na figura 4.8 mostra-se a tela principal da ferramenta.

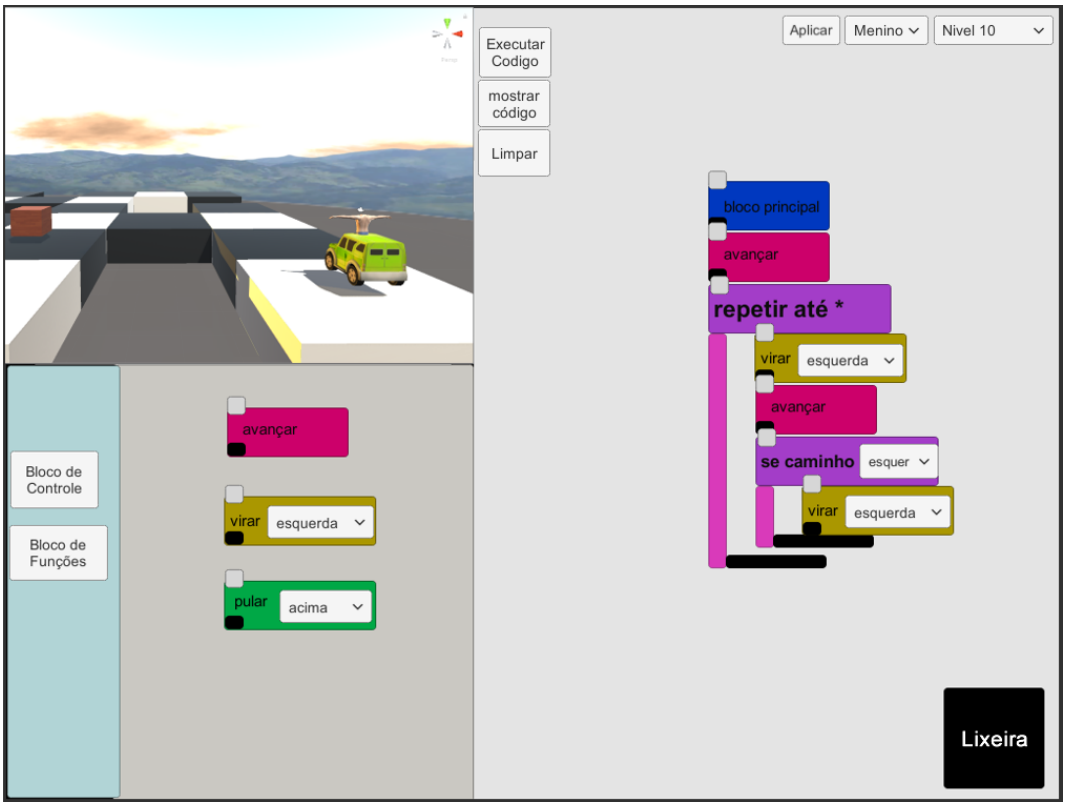


Figura 4.8 Tela principal da ferramenta virtual.

4.3.1. Interface gráfica da ferramenta

Na Figura 4.9 mostra-se a janela onde será reproduzida a animação em 3D, produto do script gerado. Esta animação é feita usando objetos do próprio Unity:

- **Cubo 3D** para a elaboração do labirinto,
- **Carro verde** é o objeto móvel que percorrerá todo o labirinto desde o início até o ponto objetivo, além de ser onde o avatar estará acima.
- **Cofre do tesouro**. É um objeto que apresenta o objetivo final à qual o carro deverá chegar, ele estará localizado acima do quadro amarelo.
- **Seta direcional** este objeto aparece quando o participante quer conhecer se tem um caminho livre para virar à esquerda, à direita, ou ir em frente, este objeto é invocado dentro da instrução condicional IF (***Sé existe caminho*** *<esquerda/direita/frente>* ***fazer alguma coisa***).
- **Audiência** formada por um conjunto de objetos pessoas que estarão localizados no ponto final junto ao cofre do tesouro.

- **Avatar** este é o objeto principal que vai representar ao participante no mundo virtual, o objeto tem a forma de humanoide e possui uma câmera no lugar de sua cabeça, isto para dar a sensação de que o corpo do objeto seja do participante. Além disso se considerou dois tipos de avatares, masculino e feminino, para que os participantes se sintam definidos e bem representados.

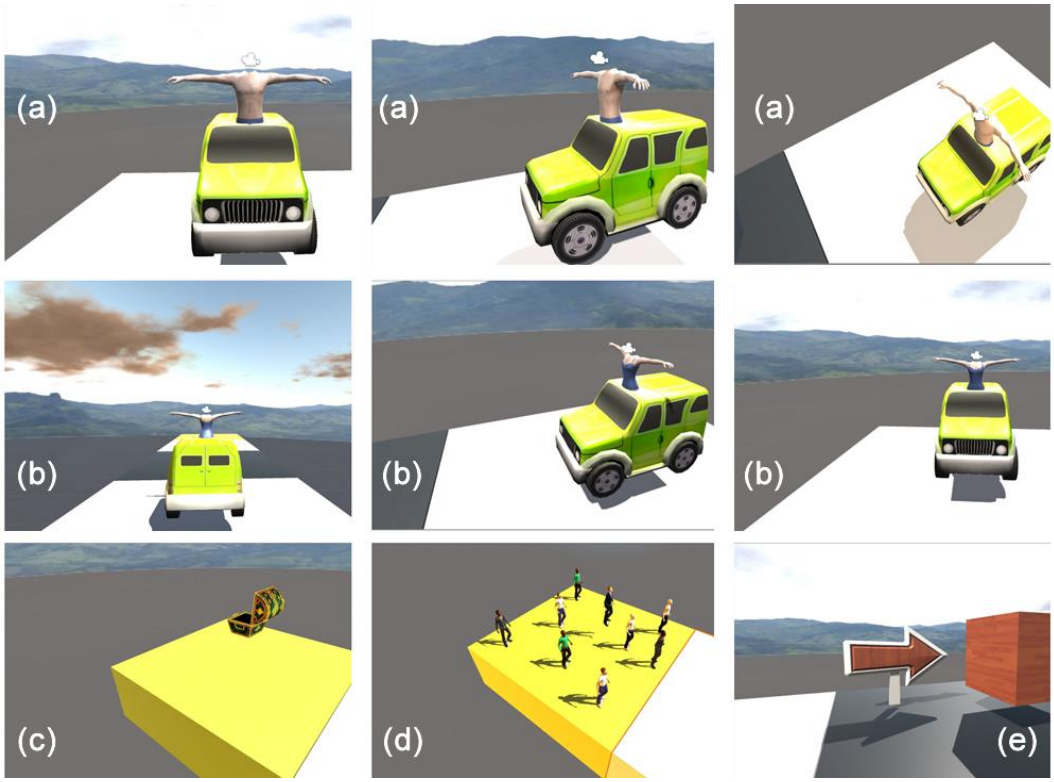


Figura 4.9(a) e (b) mostra-se ao avatar 3D para o participante (Homem e Mulher) acima de um carro. (c) Cofre do tesouro, indica o ponto final ou objetivo que deverá alcançar o participante. (d) Conjunto de pessoas que conformam a audiência. (e) Objeto seta direcional e cubo obstáculo.

Na Figura 4.10 apresenta-se a janela de Barra de opções, entre elas temos:

- **Executar código**: esta opção está encarregada de converter o código formado pelos blocos e convertê-lo em código máquina capaz de ser executado pelo compilador do Unity.
- **Mostrar código:** esta opção apresenta o código fonte do script gerado pelos blocos e é convertido em uma linguagem textual.
- **Limpar:** esta opção permite refazer todas as ações do usuário e apresenta uma janela limpa sem nenhum bloco.

- **Seletor de nível:** está opção permite selecionar o nível de desafio da ferramenta. Estes desafios estão numerados de 1 a 10
- **Seletor de gênero:** esta opção permite que o participante escolha como ele quer ser apresentado na animação: com avatar de menino ou menina.
- **Aplicar:** esta opção apresenta o cenário dependendo da escolha do participante, levando em consideração o seletor de nível e gênero.

Figura 4.10 Barra de ferramentas

Na Figura 4.11 apresenta-se a janela onde estarão os conjuntos dos blocos que serão utilizados pelos usuários. Estes conjuntos apresentam os tipos dos blocos que foram desenhados, entre eles temos: Blocos de controle e bloco de funções.

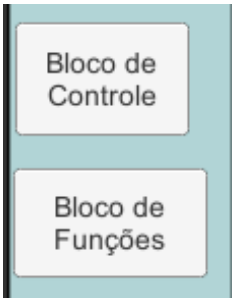


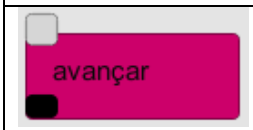
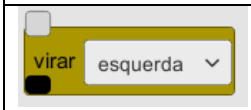
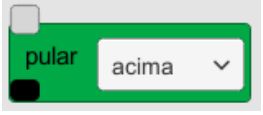
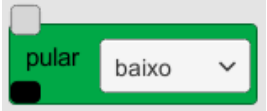
Figura 4.11 Espaço onde se apresentará o conteúdo dos conjuntos de blocos.

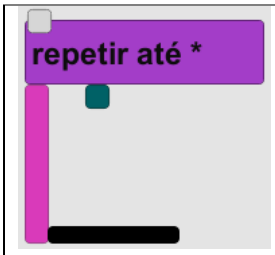
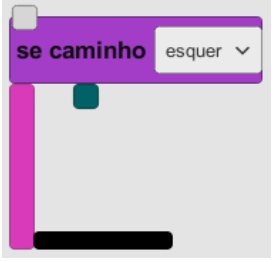
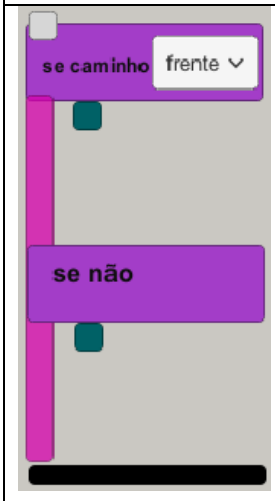
Na Figura 4.12 apresenta-se a janela onde serão colocados os blocos que servirão para formar o script de código. Esta janela tem por default o bloco principal, que servirá para que somente aqueles blocos que estejam conectados a ele possam ser convertidos em linguagem do computador e posteriormente somente este código ser executado na animação.

Figura 4.12 Espaço onde se poderá desenvolver o script.

Na Tabela 4.3 apresenta-se de forma mais detalhada os diferentes blocos da ferramenta que o usuário pode usar para resolver os desafios. Estes blocos estão divididos em funções e estruturas de controle. Entre as funções temos: *girar à esquerda*, *girar à direita*, *avançar* e entre as estruturas de controle condicional temos: *se*, *se-não e repetir*.

Tabela 4.3 Lista de blocos que formam a ferramenta de programação

Bloco	Descrição
	Este bloco faz com que o objeto siga em linha reta adiante
	Este bloco faz com que o objeto gire em 90º para a esquerda
	Este bloco faz com que objeto gire em 90º para a direita
	Este bloco faz com que o objeto pule por cima de um obstáculo.
	Este bloco faz com que o objeto pule para baixo, isto acontece sempre que o carro esteja acima do obstáculo.

	Este bloco faz a repetição das instruções internas, até encontrar o objetivo final que é o cofre de tesouro.
	Este bloco avalia se existe caminho para ir pela esquerda.
	Este bloco avalia se existe caminho para ir pela direita.
	Este bloco avalia se existe caminho para ir em frente.
	Este bloco avalia se existe caminho para ir em frente (if), em caso de não haver, faz outra ação (else).

4.3.2. Descrição da interação do usuário imerso no ambiente 3D

Como parte do estudo, se criou uma experiência de imersão para demonstrar os benefícios da realidade virtual. O principal objetivo é prover ao usuário a

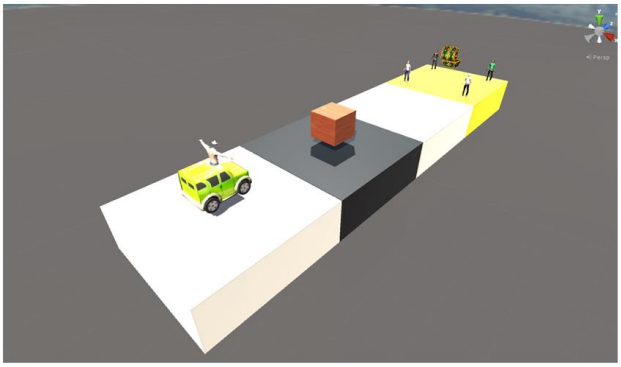
sensação de imersão, que consiste em submergir-lhe no mundo virtual, sendo ele personificado por um objeto 3D. Desta forma ele poderá participar da animação e verificar se o código gerado está bem feito ou não, isto é, se o objeto personificado chega com sucesso ao ponto final (cofre do tesouro). Ele poderá sentir a satisfação de que o script gerado por ele foi bem feito. Caso o objeto personificado não chegue ao ponto final, o Sistema apresenta uma imagem com o mensagem "Erro de código" então ele perceberá que o script gerado não é o correto e deverá corrigir o código.

Para facilitar a experiência de imersão, utilizamos o Oculus Rift DK2 (Oculus Rift 2017). O Rift tem uma resolução de 960x1080 por olho e um campo de visão diagonal de 100°. Nós também usamos o sensor de movimento Microsoft Kinect V2 (Microsoft kinect 2017) para rastrear a posição e os movimentos do usuário. A ferramenta está em execução em uma máquina com o sistema operacional Microsoft Windows 10, processador Intel Core i5, gráficos NVidia GeForce GTX 860M e 8GB de RAM.

4.3.3. Descrição dos desafios para o usuário resolver

A seguir descreve-se cada desafio da ferramenta. Cada desafio está relacionado com os *conceitos* e *comportamentos* (Concepts and Approachments) do pensamento computacional. Estes desafios têm como finalidade ensinar aos estudantes os princípios básicos da programação e a lógica. Os desafios foram selecionados da ferramenta de programação online Blockly (Blockly, 2017).

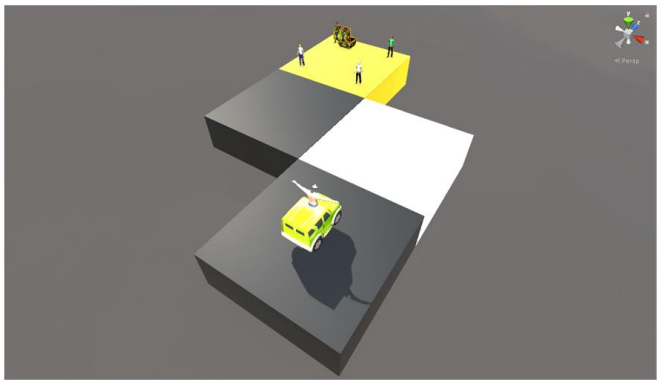
Desafio 1: Neste desafio que se mostra na Figura 4.13, o usuário deve levar o carrinho do primeiro quadro até o quadro número 3, sendo que no quadro seguinte existe um obstáculo que ele deve pular. Para isso ele deverá fazer uso dos seguintes *Concepts* (*Logic, Algorithm e Evaluation)* e os seguintes *Approaches (Tinkering, Creating e Debugging)* do pensamento computacional, e finalmente o script gerado pela ferramenta se apresenta na mesma figura.



```
//Desafio numero (1)
void animationMain() {
    jumpUp()
    moveForward();
    moveForward();
    jumpDown();
    moveForward();
}
```

Figura 4.13 Desafio número 1 da ferramenta com seu algoritmo de solução

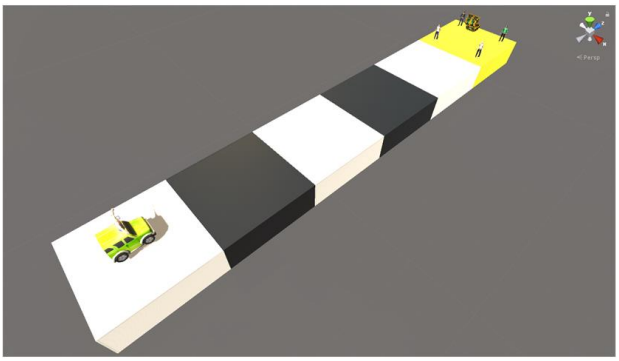
Desafio 2: Neste desafio que se mostra na Figura 4.14, o usuário deve levar o carinho do primeiro quadro para depois fazer um giro de 90 graus e continuar em linha reta por um quadro e depois girar para a direita, e finalmente chegar ao ponto final. Para isto o estudante deverá fazer uso dos seguintes *Concepts* (*Logic, Algorithm e Evaluation)* e os seguintes *Approaches (Tinkering, Creating e Debugging)* do pensamento computacional.



```
//Desafio numero (2)
void animationMain() {
    moveForward();
    turnLeft();
    moveForward();
    turnRight();
    moveForward();
}
```

Figura 4.14 Desafio número 2 da ferramenta com seu algoritmo de solução

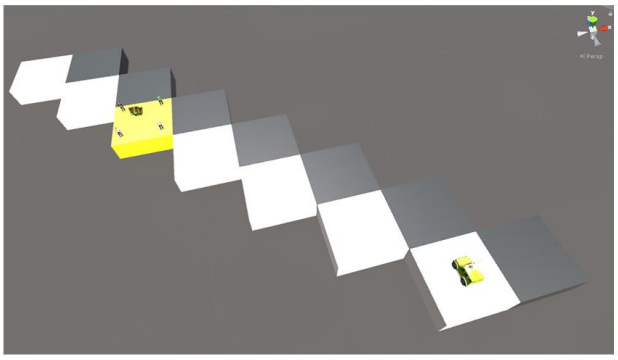
Desafio 3: Neste desafio que se mostra na Figura 4.15, o usuário deverá levar o carro em linha reta passando por 5 quadros, para isto o usuário faz uso dos seguintes *Concepts* (*Logic, Algorithm, Patterns e Evaluation)* e os seguintes *Approaches (Tinkering, Creating e Debugging)* do pensamento computacional. Neste desafio se usará a estrutura de controle repetitiva “enquanto” (while). Na mesma figura mostra-se o algoritmo na linguagem de programação C# que resolve o desafio.



```
//Desafio numero (3)
void animationMain() {
  while (notDone()) {
    moveForward();
  }
}
```

Figura 4.15 Desafio número 3 da ferramenta com seu algoritmo de solução

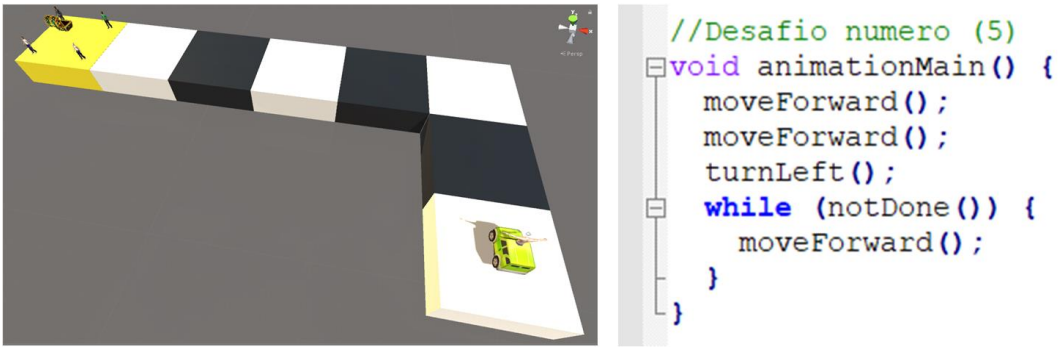
Desafio 4: Neste desafio que se mostra na Figura 4.16, o usuário deverá fazer uso da estrutura de controle repetitiva "enquanto" (while) que contém as funções usadas anteriormente. Neste desafio o usuário aprenderá sobre os seguintes *Concepts (Logic, Algorithm, Patterns e Evaluation)* e os seguintes *Approaches (Tinkering, Creating e Debugging)* do pensamento computacional. Na mesma figura mostra-se o algoritmo na linguagem de programação C# que resolve o desafio.



```
//Desafio numero (4)
void animationMain() {
  while (notDone()) {
    moveForward();
    turnLeft();
    moveForward();
    turnRight();
  }
}
```

Figura 4.16 Desafio número 4 da ferramenta com seu algoritmo de solução

Desafio 5: Neste desafio (Figura 4.17), o usuário faz uso da estrutura de controle repetitiva "enquanto" (while) e as funções usadas anteriormente. Neste desafio o usuário aprenderá os seguintes *Concepts (Logic, Algorithm, Patterns e Evaluation)* e os seguintes *Approaches (Tinkering, Creating e Debugging)* do pensamento computacional.

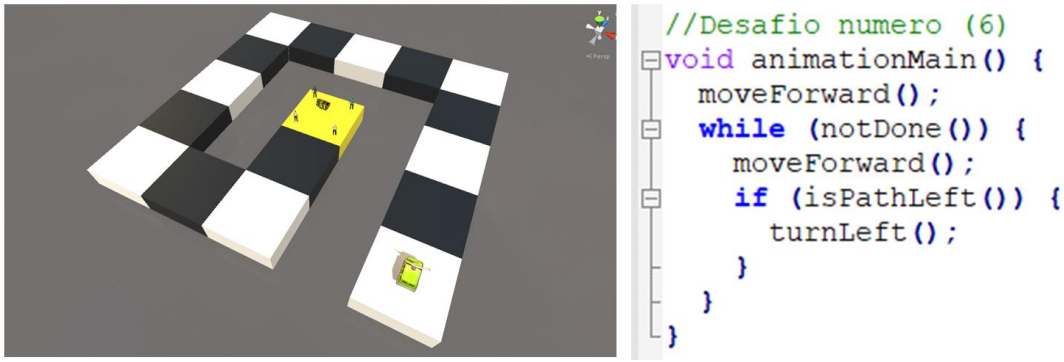


```
//Desafio numero (5)
void animationMain() {
  moveForward();
  moveForward();
  turnLeft();
  while (notDone()) {
    moveForward();
  }
}
```

Figura 4.17 Desafio número 5 da ferramenta com seu algoritmo de solução

Figura 4.17: Desafio número 5 da ferramenta com seu algoritmo de solução

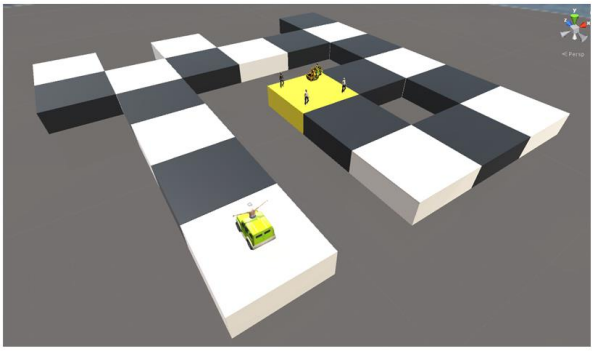
Desafio 6: Neste desafio (Figura 4.18) o usuário faz uso da estrutura de controle repetitiva "enquanto" (while) e estrutura de controle condicional “If” e as funções usadas anteriormente. Neste desafio o usuário aprenderá os seguintes *Concepts (Logic, Algorithm, Patterns e Evaluation)* e os seguintes *Approaches (Tinkering, Creating e Debugging)* do pensamento computacional.



```
//Desafio numero (6)
void animationMain() {
  moveForward();
  while (notDone()) {
    moveForward();
    if (isPathLeft()) {
      turnLeft();
    }
  }
}
```

Figura 4.18 Desafio número 6 da ferramenta com seu algoritmo de solução

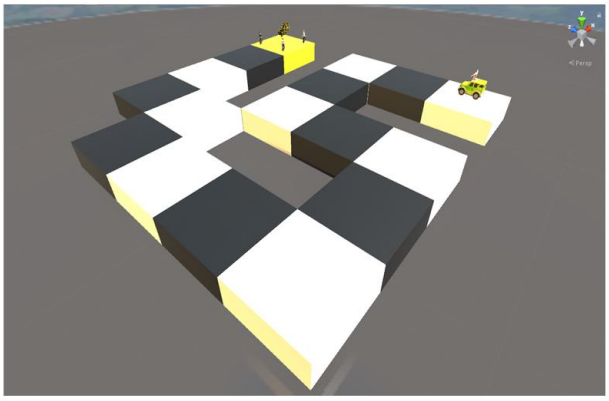
Desafio 7: Neste desafio (Figura 4.19) o usuário faz uso da estrutura de controle repetitiva “While” e estrutura de controle condicional “If” e as funções usadas anteriormente. Neste desafio o usuário aprenderá os seguintes *Concepts (Logic, Algorithm, Patterns e Evaluation)* e os seguintes *Approaches (Tinkering, Creating e Debugging)* do pensamento computacional.



```
//Desafio numero (7)
void animationMain() {
  while (notDone()) {
    if (isPathForward()) {
      moveForward();
    }
    if (isPathRight()) {
      turnRight();
    }
  }
}
```

Figura 4.19 Desafio número 7 da ferramenta com seu algoritmo de solução

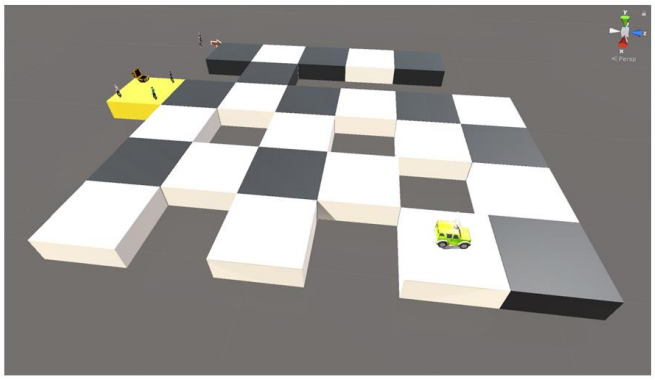
Desafio 8: Neste desafio (Figura 4.20) o usuário faz uso da estrutura de controle repetitiva "While" e estrutura de controle condicional "If" e as funções usadas anteriormente. Neste desafio o usuário aprenderá os seguintes *Concepts (Logic, Algorithm, Patterns e Evaluation)* e os seguintes *Approaches (Tinkering, Creating e Debugging)* do pensamento computacional.



```
//Desafio numero (8)
void animationMain() {
  while (notDone()) {
    moveForward();
    if (isPathLeft()) {
      turnLeft();
    }
    if (isPathRight()) {
      turnRight();
    }
  }
}
```

Figura 4.20 Desafio número 8 da ferramenta com seu algoritmo de solução

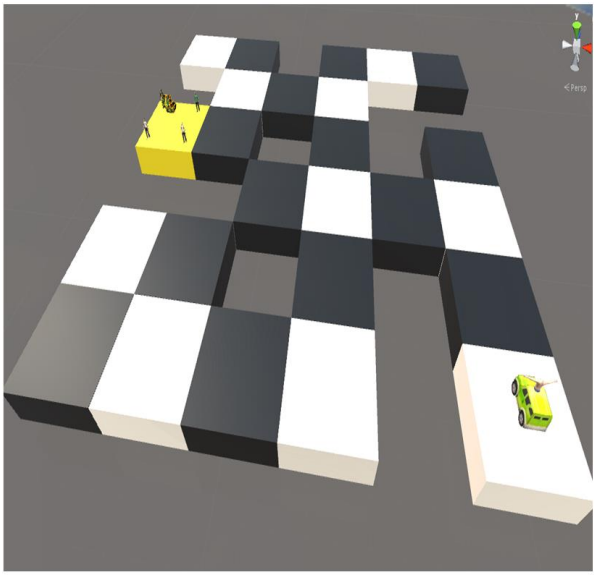
Desafio 9: Neste desafio (Figura 4.21) o usuário faz uso da estrutura de controle repetitiva "While" e estrutura de controle condicional "If" e as funções usadas anteriormente. Neste desafio o usuário aprenderá os seguintes *Concepts (Logic, Algorithm, Patterns e Evaluation)* e os seguintes *Approaches (Tinkering, Creating e Debugging)* do pensamento computacional.



```
//Desafio numero (9)
void animationMain() {
  moveForward();
  turnLeft();
  while (notDone()) {
    if (isPathForward()) {
      moveForward();
    } else {
      if (isPathLeft()) {
        turnLeft();
      }
    }
  }
}
```

Figura 4.21 Desafio número 9 da ferramenta com seu algoritmo de solução

Desafio 10: Neste desafio (Figura 4.22) o usuário faz uso da estrutura de controle repetitiva "While" e estrutura de controle condicional "If-else" e as funções usadas anteriormente. Neste desafio o usuário aprenderá os seguintes *Concepts (Logic, Algorithm, Patterns e Evaluation)* e os seguintes *Approaches (Tinkering, Creating e Debugging)* do pensamento computacional.



```
//Desafio numero (10)
void animationMain() {
  while (notDone()) {
    moveForward();
    if (isPathLeft()) {
      if (isPathRight()) {
        if (isPathForward()) {
          turnRight();
        } else {
          turnLeft();
        }
      } else {
        turnLeft();
      }
    } else {
      if (isPathRight()) {
        turnRight();
      }
    }
  }
}
```

Figura 4.22 Desafio número 10 da ferramenta com seu algoritmo de solução

4.4. Desenvolvimento do Sistema

Para começar com a implementação da ferramenta, primeiro se define a linguagem de programação visual, para esta definição será usado uma gramática de disposição de imagens (Picture Layout Grammar).

4.4.1. Linguagem de programação visual

A Gramática de Disposição de Imagens (GDI), segundo Golin et al. (1990), é um tipo de gramática que pode ser utilizada para especificar a sintaxes de linguagens visuais, da mesma forma que as gramáticas livres do contexto são utilizadas para definir linguagem de programação textual.

As linguagens de programação visual classificam-se em três grupos, de acordo as expressões visuais utilizadas: “Linguagens baseadas em ícones”, “Linguagens baseadas em formas” e “Linguagens diagrama” (Golin et al., 1990)

Nas análises da linguagem de programação visual e das linguagens de programação textual, a estrutura dos elementos que as formam são similares, já que estes estão compostos por elementos léxicos terminais e não terminais. A principal diferença é que nas linguagens de programação visual os elementos terminais no programa fonte estão ordenados em um espaço de duas dimensões, enquanto a linguagem textual tem só uma dimensão.

4.4.2. Operadores de produção pré-definidos

Na GDI as produções correspondem a operadores de composição de imagens e os atributos dos símbolos gramaticais apresentam a informação espacial para um elemento de imagem. Cada símbolo gramatical (bloco simples e bloco composto) possui 2 atributos (*male connector e female connector*), a intersecção destes atributos indicam que blocos são adjacentes.

A continuação se descrevem os dois tipos de símbolos: bloco Simples e Compostos.

- **Bloco Simples:** Está representado por um retângulo, este bloco não possui outros blocos em seu interior, para nossa ferramenta ele apresentará às funções predefinidas. Na Figura 4.23 mostra-se o bloco simples avançar.

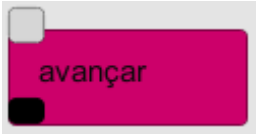


Figura 4.23 Exemplo de um bloco simples

- **Blocos compostos:** Estes blocos contêm outros sub-blocos em seu interior. Para nossa ferramenta ele apresentará as estruturas de controle condicionais (if, if-else) e de repetição (while). A Figura 4.24 mostra os blocos compostos "Repetir até" e " se ou se não".

Figura 4.24 Exemplo de um bloco composto

Os dois tipos de símbolo gramatical (bloco simples e composto) tem 2 atributos próprios: "*Male connector*" e "*Female connector*", que podem ser vistos na Figura 4.25.

O bloco composto além de ter os dois atributos (*male* e *female connector*) tem 1 ou 2 atributos internos que são "*Compound connector 1*" e outro opcional "*Compound connector 2*"

- **Male connector:** Está representado por um quadrado que está localizado na parte superior do bloco (Figura 4.25). Este *connector* permite que o bloco seja montado com outro bloco, isto é, quando o "*male connector*" esteja por cima e na mesma posição do "*female connector*" do outro bloco.
- **Female connector:** Está representado por um quadrado que está localizada na parte inferior do bloco (Figura 4.25). Este *connector* permite que os outros blocos sejam montados a ele, isto é, quando o

"*female connector*" esteja por debaixo e na mesma posição de "*male connector*" do outro bloco.

- **Compound connector:** Está representado por um quadrado localizado no interior do bloco composto (Figura 4.26). Este *connector* permite que os outros blocos sejam montados a ele pero como filhos, este *connector* tem um comportamento igual ao do "*female connector*"

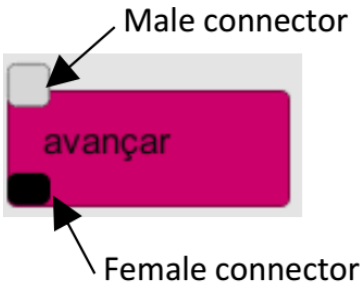


Figura 4.25 Bloco simples com seus atributos "male Connector" e "female connector"

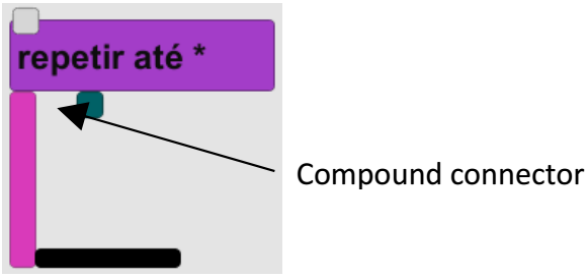


Figura 4.26 Bloco composto com o atributo "Compound connector"

Para criar as produções da ferramenta, se criaram operadores de produção predefinidos.

- **Operador Next:** Para definir que o elemento C segue depois do elemento B (Figura 4.27), se declara formalmente o operador next: $A \rightarrow next\{C, B\}$

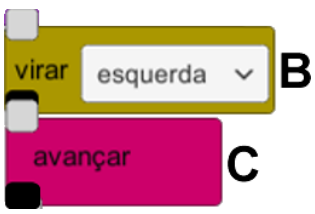


Figura 4.27 Sequência de blocos que são representados pelo operador de produção next

Onde as coordenadas de posição do "*male connector*" do Bloco C é a mesma que a posição do "*female connector*" do Bloco B:

$$C_{maleconnector}(X,Y) == B_{femaleconnector}(X,Y)$$

- **Operador Contain.** Para definir que o bloco B contém o bloco C (Figura 4.28), se declara formalmente o operador *contain*: $A \rightarrow contain\{B,C\}$

Figura 4.28 Acoplamento de blocos que são representados pelo operador de produção contain

Onde as coordenadas de posição do "*male connector*" do bloco C é a mesma que a posição do "*compound connector*" de B:

$$B_{compoundconnector}(X,Y) == C_{maleconnector}(X,Y)$$

Com a produção acima explicada, pode-se começar a criação das produções da ferramenta.

4.4.3. Regras de Produção

Na Figura 4.29 mostram-se as regras de produção que definem de maneira formal a linguagem aceita pela ferramenta:

1. $START \rightarrow next(STATEMENT, startBlock)$
2. $STATEMENT \rightarrow next(STATEMENT, BLOCK)|BLOCK$
3. $BLOCK \rightarrow turnLeftBlock|turnRightBlock|forwardBlock$
4. $BLOCK \rightarrow REPEAT|IF|IF\_ELSE$
5. $REPEAT \rightarrow contain(repeatBlock, STATEMENT)$
6. $IF \rightarrow contain(ifBlock, STATEMENT)$
7. $IF\_ELSE \rightarrow next(ELSE, IF)$
8. $ELSE \rightarrow contain(elseBlock, STATEMENT)$

Figura 4.29 Produção gerada a partir das regras definidas anteriormente

Os elementos terminais das produções estão escritos em minúscula e os elementos não terminais estão escritos em maiúscula.

Com estas produções os usuários da ferramenta poderão criar os scripts na janela de edição. Na janela de edição do compilador se poderão criar diagramas que implementam um modelo para fazer estes desenhos dirigidos pelas sintaxes, isto é, o compilador em sua interface gráfica não permite ao usuário criar sentenças que não cumpram com as regras de produção. Portanto, quando o usuário termina de realizar seu programa, só será necessário realizar a tradução para a linguagem de programação do Unity (C#). É por isso que o compilador não conta com um analisador sintático do programa fonte.

Na Figura 4.30 mostra-se um script de código que foi elaborado na ferramenta de programação, este algoritmo faz uso dos blocos simples como: *avançar e virar* e de blocos compostos como: *repetir até* e *se caminho*

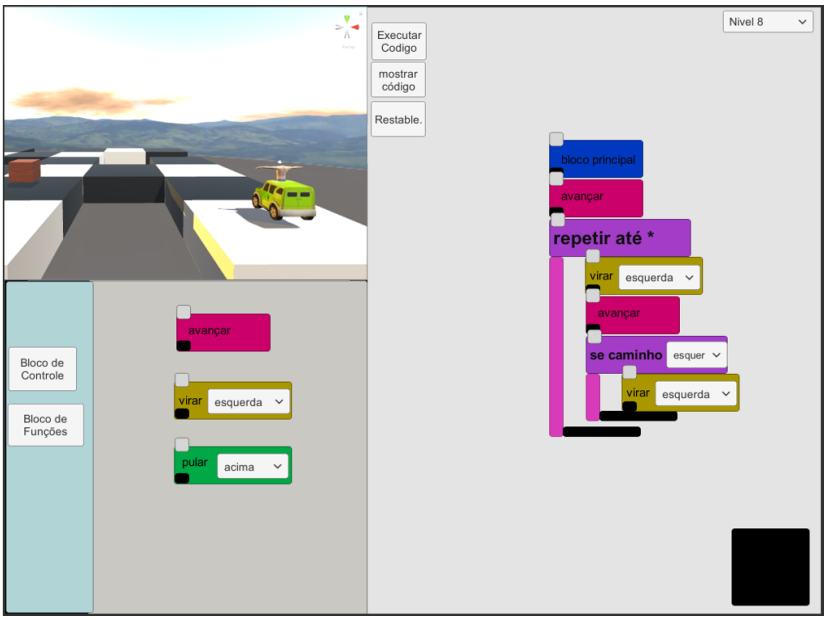


Figura 4.30 Script de código gerado pelo compilador visual.

Na Figura 4.31 mostra-se a árvore de análise sintática que foi gerada a partir do Script de código da Figura 4.30 anterior.

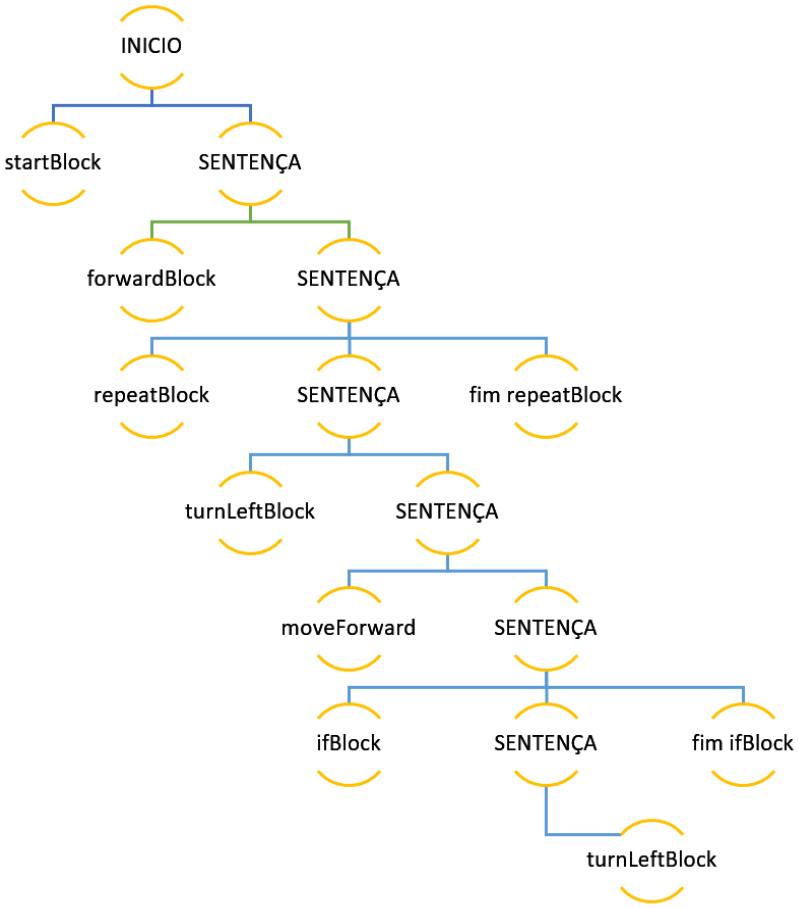


Figura 4.31 Árvore de análises sintático do script de código.

Na Figura 4.32 mostra-se o código gerado pela ferramenta de programação. Este script está feito para ser compilado e executado pelo Unity 3D (C#)

```
void animationMain(){
    moveForward();
    while (notDone()){
        turnLeft();
        moveForward();
        if(isPathLeft()){
            turnLeft();
        }
    }
}
```

Figura 4.32 Script em linguagem C# gerado pelo arvore de análise sintático.

4.4.4. Traduções

Na Figura 4.33 mostra-se o processo de tradução para o script gerado pela ferramenta visual.

Figura 4.33 Processo de tradução do script feito na ferramenta visual até obter o código em linguagem C# do Unity.

Neste capítulo definiu-se de maneira formal a linguagem de programação visual do compilador, para a qual se utilizou a gramática de disposição de imagens. Além disso, apresentou-se um exemplo de uma sentença visual e o processo de tradução à linguagem de computador.

No próximo capítulo se descreve a metodologia da avaliação usada, apresentando os participantes, o ambiente no qual os testes foram realizados, os equipamentos e materiais utilizados e o procedimento dos testes.

5
Avaliação da Ferramenta

Após a realização do processo de desenho e implementação da ferramenta, segue-se a etapa de testes com os usuários, a fim de verificar se a ferramenta desenvolvida atingiu os objetivos esperados. Ou seja, esta etapa consistiu em verificar se a ferramenta desenvolvida proporciona um ambiente no qual os estudantes se sentissem motivados por aprender mais sobre a programação. Além disso, pretendeu-se verificar o impacto do uso da tecnologia adotada, avaliando se a sensação de imersão ajuda para que os estudantes estejam motivados por cumprir todos os desafios que a ferramenta propõe.

Este capítulo descreve a metodologia da avaliação utilizada, apresentando os participantes, o ambiente no qual os testes foram realizados, os equipamentos e materiais utilizados e o procedimento dos testes em si.

5.1.
Participantes

Os participantes são estudantes de ensino fundamental da escola *Darcy Vargas* e do *Instituto Rogerio Steinberg* no Rio de Janeiro, com idades compreendidas entre os 11 e 15 anos de idade. O recrutamento dos participantes incluiu a autorização respectiva, realizado por meio do termo de consentimento apresentado no Apêndice D. Por meio deste termo, além da participação das crianças, autorizou-se a realização de fotos e filmagens para sua posterior análise.

As Tabela 5.1 e Tabela 5.2 apresentam as características gerais dos participantes do teste.

Tabela 5.1 Características gerais dos participantes, escola Darcy Vargas

Cod	Sexo	Idade	Nível de ensino fundamental
A1	M	11	6
A2	M	12	6
A3	F	14	8

A4	M	15	8
A5	M	12	6
A6	M	13	8
A7	M	13	8
A8	M	13	8
A9	M	13	8
A10	M	14	9
A11	M	15	9
A12	M	15	8

Tabela 5.2 Características gerais dos participantes, Instituto Rogerio Steinberg

Cod	Sexo	Idade	Nível de ensino fundamental
A13	M	12	7
A14	F	12	7
A15	F	13	7
A16	F	13	7
A17	M	12	7
A18	M	13	7

5.2. Local de realização dos testes

5.2.1. Escola Darcy Vargas

Esta é uma escola pública com estrutura de 6000m2 na região portuária do Rio de Janeiro. Criada em 1940, recebeu o nome de Casa do Pequeno Jornaleiro Idealizado pela primeira-dama do Brasil, esposa de Getúlio Vargas. Hoje, a fundação trabalha em parcerias com centros de ensino do Brasil e do mundo.

Os testes foram realizados no laboratório de computação desta escola, (Figura 5.1) isto por fornecer um ambiente calmo e cômodo para os participantes, provendo também um ambiente exclusivo para ensinar a matéria de computação. Este laboratório conta com computadores, uma lousa digital interativa, adicionou-se neste laboratório 2 câmeras posicionadas de forma que uma focasse em um participante, outra focasse captar a interação com o computador fazendo uso dos

Oculus Rift e Microsoft Kinect. Os testes foram realizados num período de 4 semanas. Cada sessão corresponde a um determinado nível de desafio da ferramenta. Cada sessão teve uma duração de 10 minutos no máximo. Depois da sessão, se procedia o teste do script de código gerado com os dispositivos de realidade virtual.

Em cada sessão, procurou-se oferecer auxílio aos participantes caso fosse necessário. Porém, para cada desafio, esta ajuda era diminuída, de forma que os participantes adquirissem independência ao término de cada sessão.

Figura 5.1 Estudante na sala de teste, Escola Darcy Vargas.

5.2.2. Instituto Rogerio Steinberg

O Instituto Rogerio Steinberg (IRS) é uma organização sem fins lucrativos que atua na identificação e desenvolvimento de crianças e jovens com Altas Habilidades, da cidade do Rio de Janeiro. Foi iniciado em 1998 e até hoje o Instituto, através dos seus dois programas, já atendeu cerca de 33.000 participantes.

Os testes foram realizados em uma sala especial de teste, isto por fornecer um ambiente calmo e cômodo para os participantes. Este espaço está preparado

para o trabalho individual com os alunos. Esta sala contou com uma mesa, um computador e 1 câmera posicionada de forma que focasse ao participante interatuando com o computador. Na Figura 5.2 mostra-se uma participante com os Oculus Rift e o Kinect. Os testes foram realizados num tempo de 3 horas durante um dia. Considerou-se um tempo de 5 min para cada desafio, isto porque os alunos já tinham conhecimento básicos de programação, o que ajudou muito na resolução dos desafios.

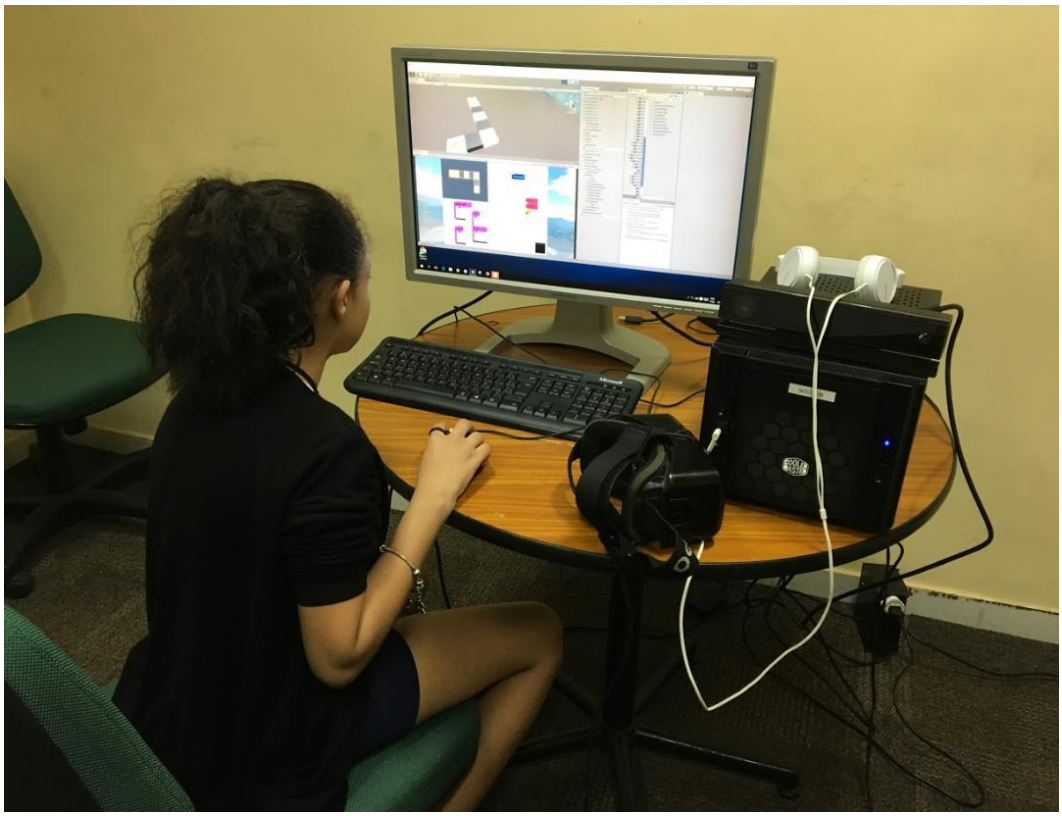


Figura 5.2 Estudante na sala de teste, Instituto Rogerio Steinberg

5.3. Procedimento

O processo de avaliação com os usuários foi realizado em três etapas: um pré-teste, teste e pós-teste. Estas avaliações foram acompanhadas do professor da matéria de computação.

5.3.1. Pré-Entrevista

Esta etapa consistiu na realização de uma entrevista antes da realização dos testes. Esta entrevista consiste em uma série de perguntas para conhecer o nível de conhecimento do participante sobre a ciência da computação, conhecer a

experiência no uso de aparelhos de Realidade Virtual, assim como, a quantidade de tempo que usa o computador e com que fim.

O questionário de perguntas para a pré-entrevista encontra-se no apêndice B

5.3.2. Testes

Nesta etapa realizou-se os testes em si. Primeiramente juntaram-se dois participantes por computador. Estes participantes teriam que resolver o desafio número 1 para conhecer as características da ferramenta e poder familiarizar-se com ela. O trabalhar junto com outro colega ajuda à interação entre eles e ajuda muito no rápido entendimento da programação. Além disso, se põe em prática uma característica da metodologia de programação *"extreme programming - Pair Programming"* de uma forma muito básica. Para a formação de equipes levou-se em conta, em primeiro lugar, a afinidade, e em segundo lugar a idade (participantes cuja idade tenham a diferença mínima devem estar juntos). Cada desafio era resolvido em uma sessão de 10 minutos no máximo; ao final da sessão os alunos teriam que testar seu código fazendo uso dos dispositivos de realidade virtual (Oculus Rift e Microsoft Kinect). Para isto, considera-se um tempo de 15 minutos como máximo para cada sessão. Ao terminar de testar todas as equipes, o pesquisador explicava o algoritmo que deu solução ao desafio, e também os conceitos e práticas do *Computational Thinking* que se utilizou para resolvê-lo. Com isto asseguramo-nos que os estudantes entendiam claramente como deveriam resolver os desafios, e ficariam mais habilitados para resolver o desafio seguinte.

5.3.3. Pós-Entrevista

Após a realização dos testes, foi feita novamente uma entrevista com os estudantes. Nesta entrevista foram consideradas perguntas para conhecer o impacto que a ferramenta causou e para conhecer como foi sua primeira reação com a realidade virtual. Além disso, a entrevista também visava conhecer sua opinião sobre a ferramenta virtual (coisas que gostaram e que não gostaram), e que coisas poderiam melhorar, remover ou adicionar. Também foram considerados perguntas próprias para avaliar a imersão no mundo virtual, tendo como finalidade medir a sensação de *Telepresença*, refere-se à sensação de estar

presente no ambiente virtual, isto é, medir esse senso de "estar lá". *Presença Social*, refere-se a quanto um corpo se sente imerso e representado no ambiente virtual e com habilidades para se conectar com outros elementos do ambiente e *Usabilidade, satisfação e entusiasmo*, refere-se a conhecer a aceitação da ferramenta de programação e da experiência de imersão no mundo virtual (todas as perguntas do questionário pós-teste estão no apêndice C).

Esta entrevista tem perguntas como: De uma forma geral, o que você achou do desempenho dos participantes no teste? Como você acha que foi o desempenho no primeiro nível para cada participante? Como foi o primeiro contato dele com a tecnologia de realidade Virtual? Houve dificuldades no entendimento da atividade? Você acha que de alguma forma o uso das novas tecnologias de Realidade Virtual, promoveu o interesse deles em aprender a programar?

6
Resultados

Este capítulo descreve a etapa de testes realizada com os participantes da escola Darcy Vargas e do Instituto Rogerio Steinberg do Rio de Janeiro.

Serão apresentados os resultados observados durante as avaliações, bem como os resultados colhidos nos formulários pré-teste e pós-teste. Ao final se apresentará os comentários gerais sobre o desempenho dos usuários na ferramenta.

Como se explicou anteriormente os alunos responderam dois questionários: o questionário pré-teste e o questionário pós-teste, alguns critérios da avaliação (Perguntas e opções de resposta) foram tomadas do trabalho de Parmar D. et (2016). O primeiro questionário nos permitiu entender sobre a experiência no uso dos dispositivos electrónicos (Computadores e aparelhos de RV), além o conhecimento dos participantes sobre os conceitos básicos da programação.

A seguir faremos uma comparação entre os participantes destas duas escolas levando em consideração as perguntas do questionário pré-teste. Para isso começaremos com a pergunta. Com *que dispositivo (s) você interage?* Pode-se ver a Tabela 6.1 que representa os resultados da escola Darcy Vargas e o IRS, nele pode-se notar que 100% (n=6) dos participantes do IRS interage mais com dispositivos *Computador Notebooks* e *Smartphones*, como também os participantes da escola Darcy Vargas eles usam os dispositivos em um 100% (n = 12) e um 92% (n = 11) respectivamente, além disso um dos participantes do IRS expressou que também fazia uso de *outros dispositivos* como os *consoles de Video Games*. Pode-se destacar desta pergunta que a maioria de participantes tem uma experiência no uso do computador e o Smartphone, isto é positivo para a avaliação já que permite uma comunicação mais fluente a respeito do vocabulário que será usado durante a fase do teste.

Tabela 6.1Resultados da pergunta: Com que dispositivo (s) você interage?

Com que dispositivo (s) você interage?	Darcy Vargas (12 participantes = 100%)	IRS (6 participantes = 100%)
Caixas eletrônicos	25%	33%
Computador Desktop	33%	17%
Computador Notebooks	100%	100%
Smartphone	92%	100%
Tablet	67%	50%
Outro(s)	0%	17%

Agora as seguintes perguntas foram contestadas por todos os participantes que marcaram a alternativa computador (Desktop ou Notebook) da pergunta anterior. As perguntas são: *Frequência de utilização do computador? e Com que fim usa o computador?*. Pode-se ver na Tabela 6.2 e Tabela 6.3 as respostas a estas perguntas respectivamente. Nela pode notar o Seguinte: O 92% (n = 11) dos participantes da escola Darcy Vargas e o 67% (n = 4) dos participantes do IRS usam o computador pelo menos uma vez ao dia, isto demonstra que eles interagem com o computador muito frequente, posteriormente a tabela 6.3 mostra a informação em relação à pergunta. Com *que fim usa o computador?* Para esta pergunta se considerou uma múltipla escolha das alternativas. Pode-se notar que as duas atividades que realizam em sua maioria são *Jogo de entretenimento* e *Internet*. Além alguns participantes do IRS expressaram que eles usavam o computador para "*Fazer trabalhos escolares e assistir Aulas de robótica e codificação*".

Com esta pergunta reforçamos nossa ideia que os estudantes estão muito familiarizados com computadores e, além disso, alguns deles já têm uma experiência em relação à programação.

Tabela 6.2 Resultados da pergunta: Frequência de utilização do computador?

Frequência de utilização do computador?	Darcy Vargas (12 participantes = 100%)	IRS (6 participantes = 100%)
Pelo menos uma vez por dia	92%	67%
Pelo menos uma vez por semana	8%	33%
Raramente	0%	0%
Nunca	0%	0%

Tabela 6.3 Resultados da pergunta: Com que fim usa o computador?

Com que fim usa o computador?	Darcy Vargas (12 participantes = 100%)	IRS (6 participantes = 100%)
Jogos e entretenimento	67%	67%
Internet	92%	83%
Aplicativos e ferramentas de desenho	17%	50%
Outro (s)	25%	50%

Para esta pesquisa é muito importante conhecer se os participantes já tiveram alguma experiência no uso de dispositivos de Realidade Virtual, para isso se fez as perguntas: *O que dispositivos de Realidade Virtual você utilizou? Qual (s) tipo (s) de aplicação de RV (3D) você utiliza? e Qual é sua experiência utilizando aplicações de realidade virtual?,* nesta pergunta o participante pode fazer uma múltipla escolha das alternativas. A tabela 6.4 mostra a resposta dos participantes da Escola Darcy Vargas e o IRS, pode-se notar que a maioria de participantes 83% (n=10) e 67% (n=4) respectivamente nunca tiveram a oportunidade de usar os dispositivos de realidade virtual, e os participantes que se usaram estes dispositivos como o PlayStation VR (8% e 33%) expressaram que o usaram para os videogames, além disso consideram sua experiência no uso de aplicações de RV como pouca.

Desta pergunta destacamos que a maioria dos participantes não conhece os dispositivos de RV. A ferramenta de programação virtual servirá para que eles

tinham sua primeira experiência em RV, além disso eles usaram a ferramenta com fins educativos mediante a solução dos desafios formando script de código.

Tabela 6.4 Resultados da pergunta: Quais dispositivos de Realidade Virtual você utilizou?

Quais dispositivos de Realidade Virtual você utilizou?	Darcy Vargas (12 participantes = 100%)	IRS (6 participantes = 100%)
Oculus Rift	8%	17%
HTC Vive	0%	0%
Google Cardboard	0%	0%
Google Gear VR	0%	17%
PlayStation VR	8%	33%
Nenhum	83%	67%

A seguir descrevemos os resultados das 3 perguntas: *O que é programar? O que é um algoritmo? e O que os cientistas da computação fazem?*, que permitirá saber se eles conhecem os conceitos básicos da programação e se já tiveram alguma experiência na programação e elaboração de algoritmos, estas perguntas foram realizadas no pré-teste e pós-teste. Na Figura 6.1 pode-se ver os estudantes respondendo o questionário antes de começar a avaliação no IRS.

Figura 6.1 Participantes respondendo às perguntas do questionário pré-teste

Na Figura 6.2 mostra-se o histograma que se obteve depois de fazer a seguinte pergunta *O que é programar?* Nesta figura podemos ver que a pergunta que foi feita no questionário pré-teste (coluna de cor azul) um total de (n=6) alunos marcaram a opção *é reparar o computador*, depois (n=5) alunos marcaram *é usar as aplicações do computador word, facebook, twitter, email, etc.* (n=3) alunos marcaram a opção *não tenho ideia* e só (n=4) alunos marcaram a opção *é escrever instruções para que o computador o faça*, que é a mais acertada a respeito da atividade dos cientistas da computação.

Desta pergunta podemos deduzir que só 22% de alunos tinham uma noção de que é programar. Posteriormente esta mesma pergunta foi feita, no final da avaliação, e o resultado foi muito satisfatório sobre o que agora estes alunos sabem sobre o que é a programação. Pode-se ver que (n=17), ou 94% dos participantes, escolheram a opção correta (coluna de cor laranja).

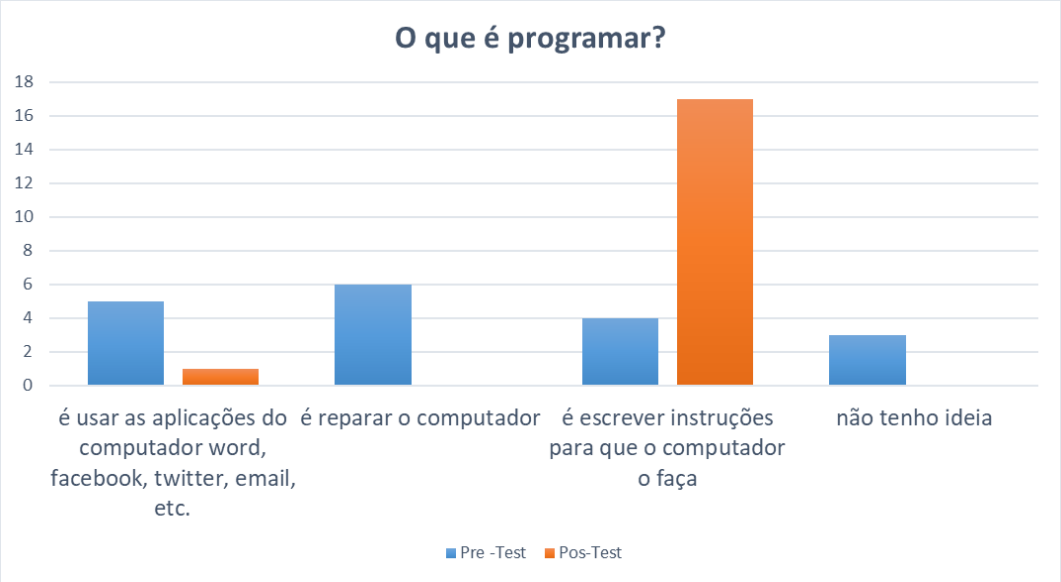


Figura 6.2 Resultado da apresentação das análises temáticas na pré-teste e pós-teste do questionário em "O que é programar? "

Na Figura 6.3 mostra-se o histograma que se obteve depois de fazer a seguinte pergunta *O que é um algoritmo?* Nesta figura podemos ver que a pergunta que foi feita no questionário pré-teste (coluna de cor azul) um total de (n=7) alunos marcaram a opção *uma parte do computador*, depois (n=6) alunos marcaram *não tenho ideia*, (n=2) alunos marcaram a opção *uma receita de cozinha* e só (n=3) alunos marcaram a opção *um conjunto de passos ordenados para resolver algo* que é a correta.

Desta pergunta podemos deduzir que só 16% de participantes tinham uma noção sobre o que é um algoritmo. Posteriormente esta mesma pergunta foi feita, depois da avaliação, e o resultado foi muito satisfatório sobre o que agora estes alunos sabem sobre o algoritmo. Pode-se ver que (n=18), isto é, 100% dos participantes, escolheram a opção correta.

Figura 6.3 Resultado da pergunta "O que é um algoritmo" do questionário pré-teste e pós-teste.

Na Figura 6.4 mostra-se o histograma que se obteve depois de fazer a seguinte pergunta *O que os cientistas da computação fazem?*, Nesta figura podemos ver que a pergunta que foi feita no questionário pré-teste (coluna de cor azul) um total de (n=5) alunos marcaram a opção *Criam Software*, depois (n=4) alunos marcaram *Fazem/Consertam/Criam Computadores*, (n=3) alunos marcaram a opção *Escrevem Código/Programas*, (n=1) alunos marcaram a opção *Fazem filmes/Videogames/Animações* e (n=5) alunos marcaram a opção *não tenho ideia*.

Desta pergunta podemos deduzir que só 44% dos participantes tinham uma noção sobre o que um cientista de computação faz. Estes participantes marcaram uma das duas opções: *criam software* ou *escrevem código/programas*, sendo estas opções as atividades mais comuns dos cientistas. Posteriormente esta mesma pergunta foi feita e o resultado foi muito satisfatório sobre o que agora estes

participantes sabem dos cientistas de computação, pode-se ver que (n=14), isto é, um total de 77% dos participantes, escolheram dentre as duas opções corretas.

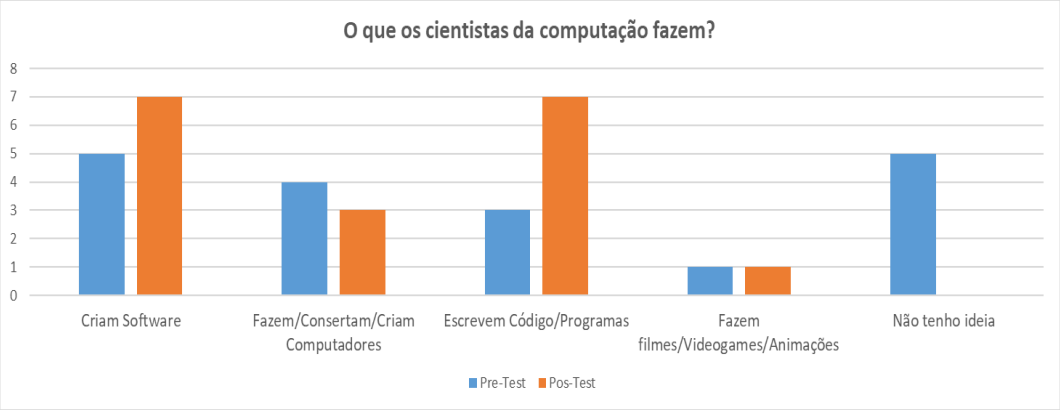


Figura 6.4 Resultado da pergunta "O que os cientistas da computação fazem? " Do questionário pré-teste e pós-teste.

6.1. Resultados esperados sobre o conhecimento da programação.

Pode-se observar na Figura 6.5 a formação das duplas de participantes por computador. Esta forma de trabalhar em pares permitiu que eles compartilhassem o conhecimento e facilitou resolver a maior quantidade de desafios.

Figura 6.5 Participantes do IRS(esquerda) e Darcy Vargas(direita) resolvendo os desafios em duplas.

Na seguinte Tabela 6.5 pode-se ver qual foi a média de tempo que levou para que os estudantes resolvam cada desafio, o tempo foi medido em minutos. Com esta informação podemos desenhar um histograma para que mostra como foi o desempenho de cada escola para a solução dos desafios. Na figura 6.5 mostra-se a informação dos participantes da escola Darcy Vargas e IRS respectivamente.

Tabela 6.5 Média de tempo que tomou para resolver cada desafio

N° de Desafio	IRS	Escola Darcy Vargas
Desafio 1	2.7	3.4
Desafio 2	3.3	3.4
Desafio 3	3.6	3.8
Desafio 4	3.4	3.9
Desafio 5	3.9	4.5
Desafio 6	4.4	6.6
Desafio 7	4.4	6.9
Desafio 8	4.5	7.4
Desafio 9	4.7	8.3
Desafio 10	5.0	-

Nesta figura 6.6 pode-se ver o histograma do desvio padrão que resultou dos tempos que levaram os participantes da escola Darcy Vargas para solucionar cada desafio. Além disso, pode-se notar a dificuldade que eles tiveram durante a solução dos desafios avançados, isto no desafio 7, 8, e 9. Também que só 8 deles conseguiram resolver o desafio n° 8 e 10 deles conseguiram resolver o desafio n° 9, e nenhum dos participantes conseguiu resolver o desafio n° 10

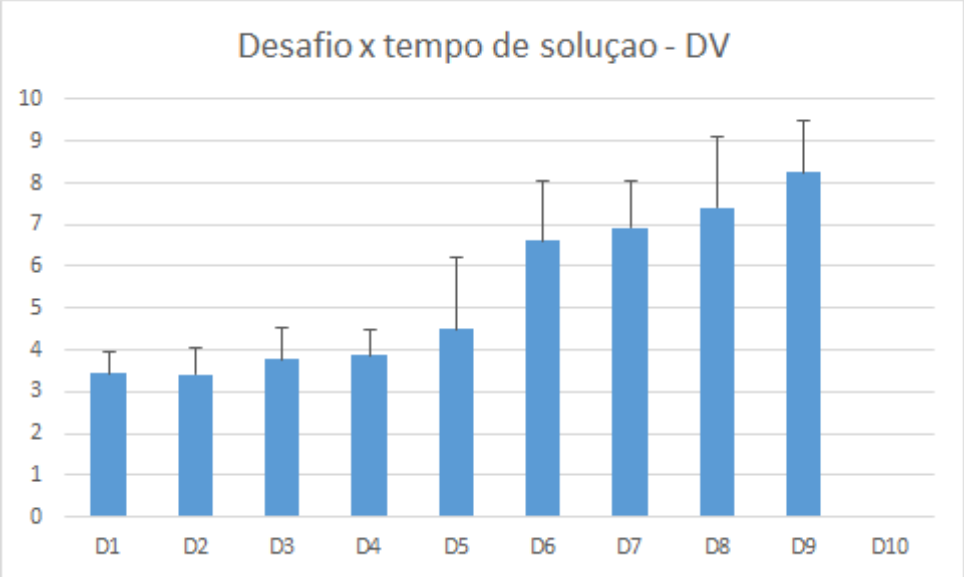


Figura 6.6 Desvio padrão do tempo que demorou o grupo da escola Darcy Vargas para cada desafio

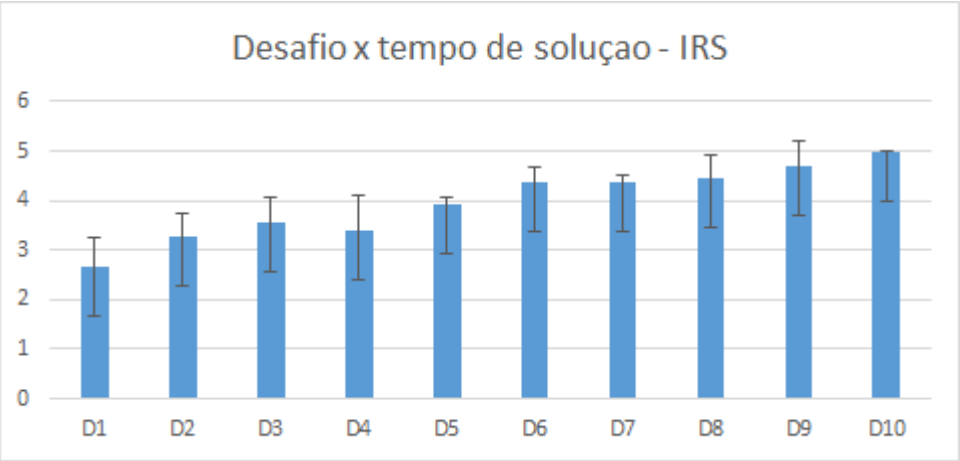


Figura 6.7 Desvio padrão do tempo que demorou o grupo do IRS para cada desafio

Sobre o pensamento computacional, pode-se explicar que os participantes usaram os conceitos e comportamentos de uma forma indireta, isto é, aplicando o processo de solução a todos os desafios. Este processo consiste em que os participantes primeiro analisaram o desafio, posteriormente propuseram uma solução usando os algoritmos de construção, e depois verificaram se a solução é funcional no ambiente virtual. Em caso de ter algum erro, teriam que encontrar o erro e resolvê-lo, e posteriormente propor alguma melhora do código fonte (caso seja possível otimizar o código levando em conta o tempo de chegada ao objetivo e a menor quantidade de blocos). A ferramenta fornece todas as funcionalidades que se necessita para realizar todo o processo de elaboração de código fonte. Dentre estas funcionalidades encontra-se *depurar o código fonte*, isto é, ele permitiu encontrar o bloco de código que ocasionava o problema e facilitou sua correção. Além disso, para a elaboração do código fonte eles primeiro tinham que aplicar a lógica de programação, posteriormente para alguns desafios eles usaram estruturas de controle repetitivas, isto é, reutilizar alguns dos procedimentos repetidos como padrões. Por exemplo, no caso do desafio número 4 da Figura 6.8, a maioria dos participantes, cerca de 78% (n=14), desenvolveram o algoritmo com a primeira Figura 6.9(a), isto é, usando 16 blocos, os outros 4 alunos deram como solução a Figura 6.9(b), isto é usando 5 blocos, e nenhum dos estudantes conseguiu resolver o desafio como apresenta a terceira Figura 6.9(c). Pode-se notar que a maioria de estudantes preferiu repetir o código fonte, mas depois de que o guia mostrou para eles a terceira solução, isto é, usando dentro de uma

estrutura de controle repetitivas o bloco "Avançar", e uma condicional se-senão e dentro dela os blocos "virar esquerda e direita", alguns dos estudantes ficaram assombrados sobre esta nova forma de programação. Este exemplo ajudou a prepará-los para os desafios posteriores.

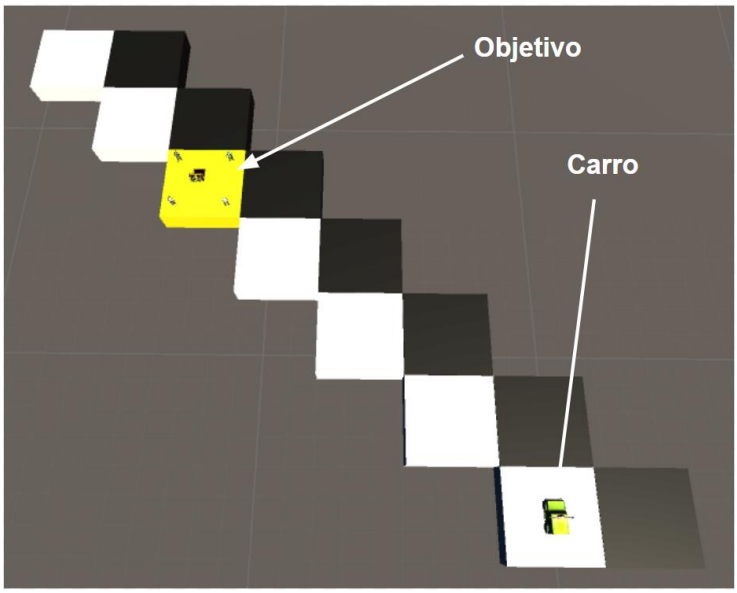


Figura 6.8 Desafio número 4 da ferramenta virtual

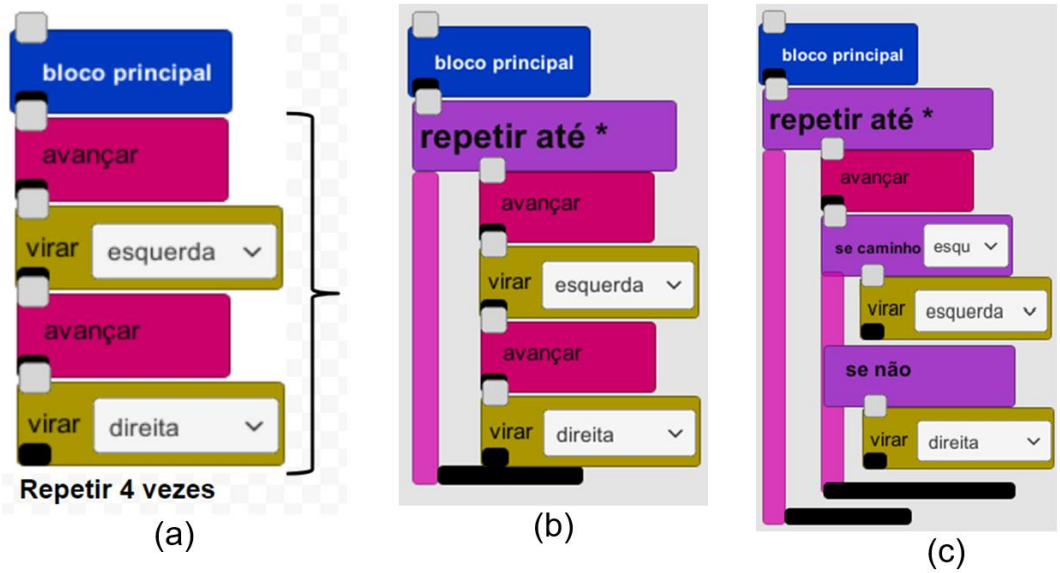


Figura 6.9 Possíveis soluções para o desafio número 4

Pode-se ver na Figura 6.10 a quantidade de estudantes que conseguiram resolver os desafios.

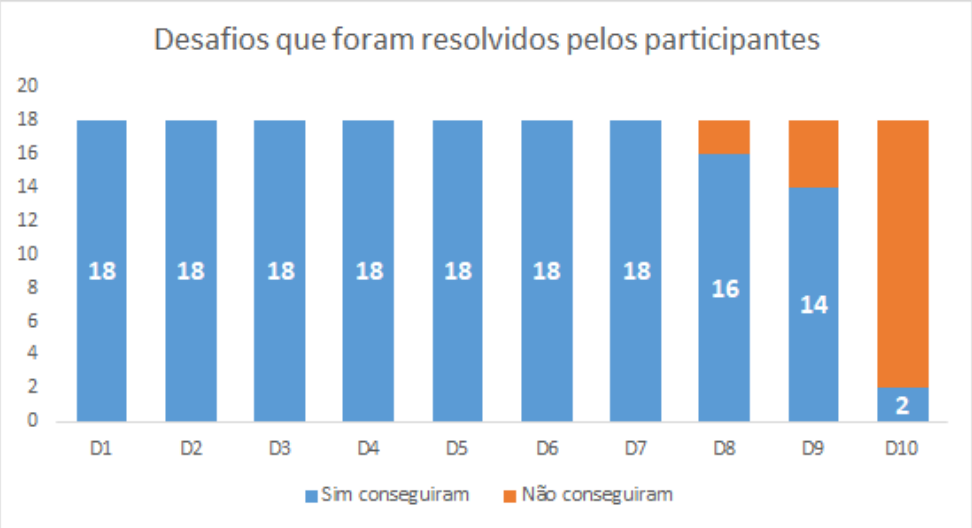


Figura 6.10 Histograma que mostra a quantidade de desafios que conseguiram e não conseguiram resolver os participantes

Do total dos 10 desafios que foram propostos, 100% (n=18) dos participantes conseguiu resolver os primeiros 7 desafios. Destes 7 desafios os primeiros 3 foram resolvidas com ajuda de um guia, podendo ser o professor ou o pesquisador; os outros 4 desafios foram resolvidos por eles mesmos. Cerca de 89% (n=16) dos participantes conseguiram resolver o desafio número 8, enquanto 78% (n=14) deles conseguiram resolver o desafio número 9. Se observou que o principal problema que eles tiveram nestes desafios foi o uso da estrutura de controle condicional “if-else”, apresentado pelo bloco composto "se, fazer alguma acção; senão, fazer outra ação". Cabe ressaltar que só dois participantes conseguiram resolver o desafio número 10 da Figura 6.11(a). Os outros participantes propuseram soluções que não conseguiam fazer com que o carro chegasse ao ponto final. Pode-se ver na Figura 6.11(b) uma solução proposta pelo par de estudantes que foram os únicos que resolveram o desafio, nesta proposta de solução eles descobriram que sua proposta fazia com que o carro percorresse todo os caminhos, isto é, não seguiu o caminho mais curto para chegar ao objetivo, mas eles se justificaram pela quantidade de blocos utilizados para a solução (7 blocos), já que isto é menor que a solução esperada (10 blocos) da Figura 6.11(c). Esta situação ajudou muito a discernir qual das duas soluções é a melhor para resolver o desafio, isto é, a fazer uma tomada de decisão entre qual é mais importante: fazer uma escolha dentre a menor quantidade de blocos ou o menor caminho para encontrar o objetivo. Então eles começaram a discutir sobre a melhor solução e

como resultado desta discussão a maioria de estudantes cerca de 89% (n=16) deles considerou optar pela rota mais curta. A explicação que eles deram foi que a pessoa montada no carro precisa chegar no menor tempo possível a seu objetivo, eles associaram este caso a suas atividades de transporte da vida diária.

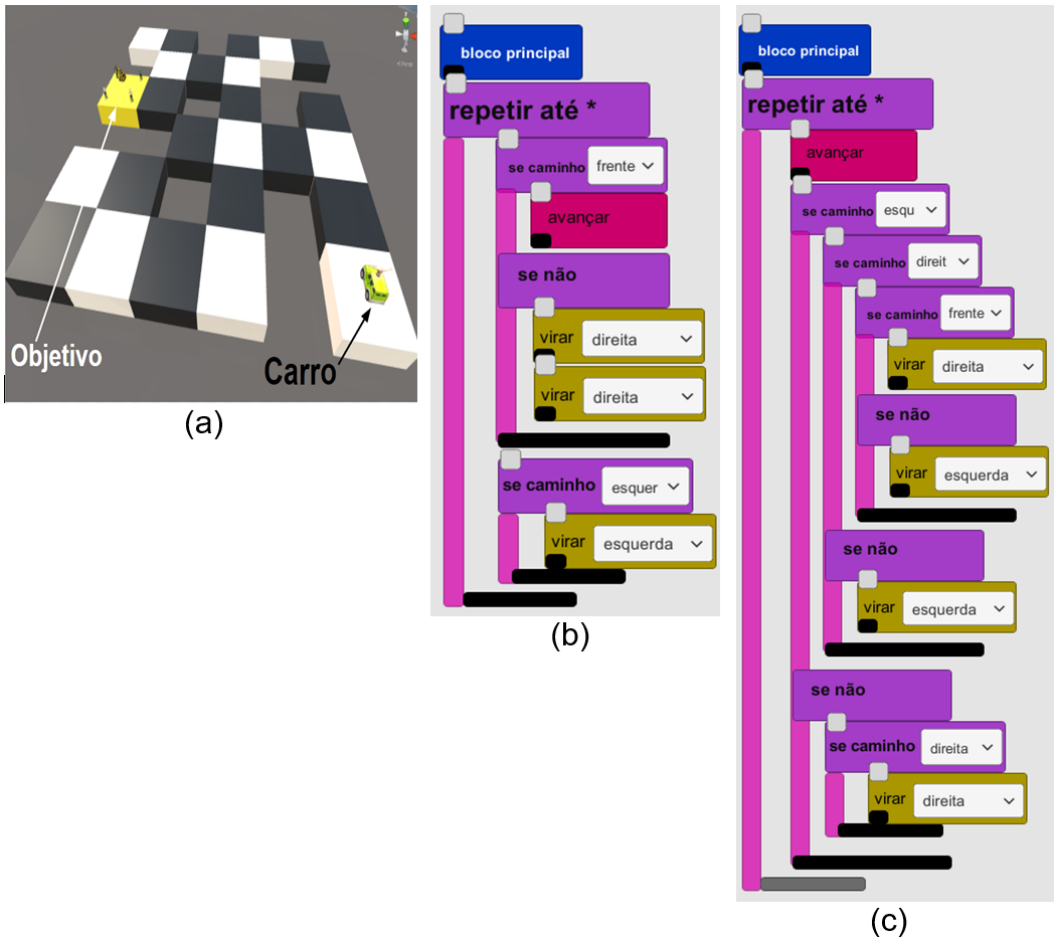


Figura 6.11 Desafio n°10 e Scripts diferentes de código feito na ferramenta para resolvê-lo.

6.2. Resultados da interação no mundo virtual

Para fazer a medição dentro do ambiente virtual, fizemos um estudo capaz de medir os princípios básicos de uma ferramenta de Realidade Virtual o que é: “Telepresença, Presença Social e Usabilidade”, as perguntas do questionário para avaliar estas sensações foi obtido do trabalho feito por Dhaval Parmar et al. (2016).

6.2.1.
Resultados em Telepresença

A Telepresença refere-se à sensação que tem os participantes de encontrar seu corpo dentro de um mundo virtual, isto é, medir esse senso de "estar lá". A ferramenta de programação oferece uma experiência de imersão no entorno 3D, isto depois de que os participantes desenvolvam seus desafios e comecem a avaliar o script de código que eles fizeram. Para medir este senso se realizou as seguintes perguntas. Você *sentiu que estava dentro e cercado pelo meio ambiente*?", e "*Como se sentiu vendo seu próprio corpo lá?*"

Na pergunta "como se sentiu ao ver seu próprio corpo lá?", as respostas de alguns participantes foram:

"Foi muito legal e também interessante...", "Foi uma experiência muito legal :) ... ", "surpreso algo estranha...", "Eu me senti dentro do jogo, Muito legal, gostei muito", "Foi muito importante, me senti feliz por ter tido a experiência de me ver lá...", "Foi ótimo, senti uma altura tipo que ia cair e tinha que observar as pessoas que teve batendo palmas ..."

Na pergunta "Você sentiu que estava dentro e cercado pelo meio ambiente?", as respostas de alguns participantes foram:

"Sim!!!!!", "não", "sim, era tudo muito real…", "sim, parecia que eu estava dentro do jogo…", "Sim estava preso e no final tinha uma estrada aberta e dá impressão que vai cair..."

Cabe ressaltar que nem todas as respostas foram positivas para a experiência de imersão. Na entrevista, cerca de 20% dos participantes fizeram comentários negativos, tais como: *"as pessoas no lugar não são muito reais ...", "A água não se mexe como numa lagoa real", "O carro está andando muito devagar".*

Destas respostas frente à avaliação de Telepresença, pode-se resumir que a ferramenta deu para os participantes novos sentimento como a emoção e o realismo além de ser divertida. Além alguns participantes tiveram algumas complicações respeito ao fone de ouvido, isto porque o cabo não tinha o tamanho necessário que permitirá que o participante se mexera em sua cadeira, também que algum deles encontrou que os personagem da audiência não tinham muita aparência real. Estes comentários serão tratados posteriormente dentro dos trabalhos futuros.

6.2.2.
Resultados em presença social

A presença social refere-se, a quanto um corpo se sente imerso e representado no ambiente virtual e com habilidades para se conectar com outros elementos do ambiente. A ferramenta permitiu ao o participante escolher entre dois tipos de avatares (homem e mulher), este avatar vai estar montado acima do carro, além um conjunto de pessoas que conformam a audiência nossa ideia era que o personagem se sinta ligado com seu avatar, além disso esperávamos que o participante se sinta motivado pela audiência e que possa interagir com os movimentos de seu braços e pernas. Para medir o sentimento de enlace social dos alunos com o personagem virtual, fizemos perguntas como: *Como sentiu-se quando era incentivado pelo público?*, *O personagem se sentiu real?* e *A pessoas que aparecem na ferramenta parecem reais?*

Na Figura 6.12 pode-se ver o resultado obtido na pergunta *Como sentiu-se quando era incentivado pelo público?*,O participante tinha a opção de escolher diferentes tipos de sentimiento, do histograma ressaltamos que a maioria de participantes sentiram-se motivados e animados..

Figura 6.12 Histograma de opções escolhidas pelos participantes

Na Figura 6.13 pode-se ver o histograma gerado a partir das respostas das perguntas de presença social dos participantes. Na pergunta “O personagem se sentiu real? ” As respostas foram um total de 92% dos participantes escolheram

(sim), e os outros escolheram (não). Na pergunta “A pessoas que aparecem na ferramenta parecem reais? ” as respostas foram um total de 69% dos participantes escolheram (sim) e os outros (não)

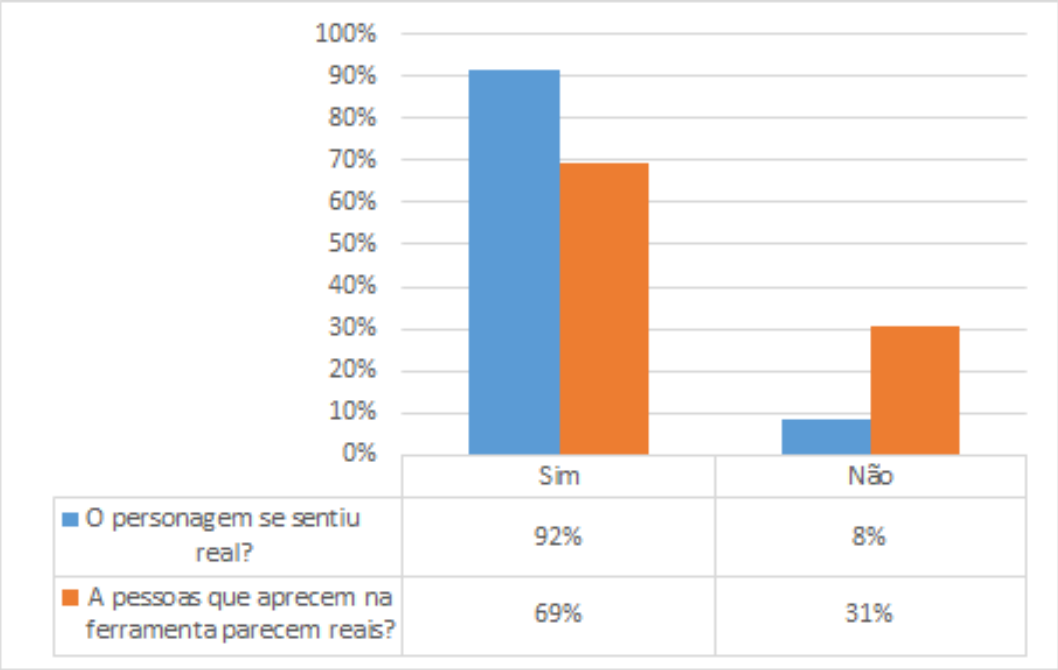


Figura 6.13 Desenho de Histograma que apresenta as respostas dos alunos a perguntas de avaliação de Presença Social

6.2.3. Resultados em Usabilidade, Satisfação e Entusiasmo

A perguntas feitas com respeito à Usabilidade Satisfação e Entusiasmo nos permitiu conhecer se os participantes estiveram satisfeitos e entusiasmados com o uso da ferramenta virtual de programação, para isto se fizeram as seguintes perguntas: “Quão provável que você use esse sistema? O que você mudaria nesse sistema? e Você acha que essa experiência imersiva o ajudaria a aprender melhor?”. As respostas dos participantes ajudaram-nos no futuro a seguir com o desenvolvimento da ferramenta.

As respostas que os alunos deram para a pergunta “O que você mudaria nesse sistema? ” São: *“nada”, “Uns gráficos e Só...”, “Iria fazer com que ele ficasse mais real...”, “Eu colocaria uma coisa, mas espontânea...”, “Os bonecos...”, “foi muito legal nunca tinha visto um daquele”, “o foco dos óculos”* e *“nada só a cor para cada tipo de pessoa”*. E para a pergunta de *“Quão provável é que você use esse sistema? ” e “Você acha que essa experiência*

imersiva o ajudaria a aprender melhor?" Desenhou-se um histograma (Figura 6.14) sobre a escolha dos participantes entre as opções de "muito, pouco ou nada"

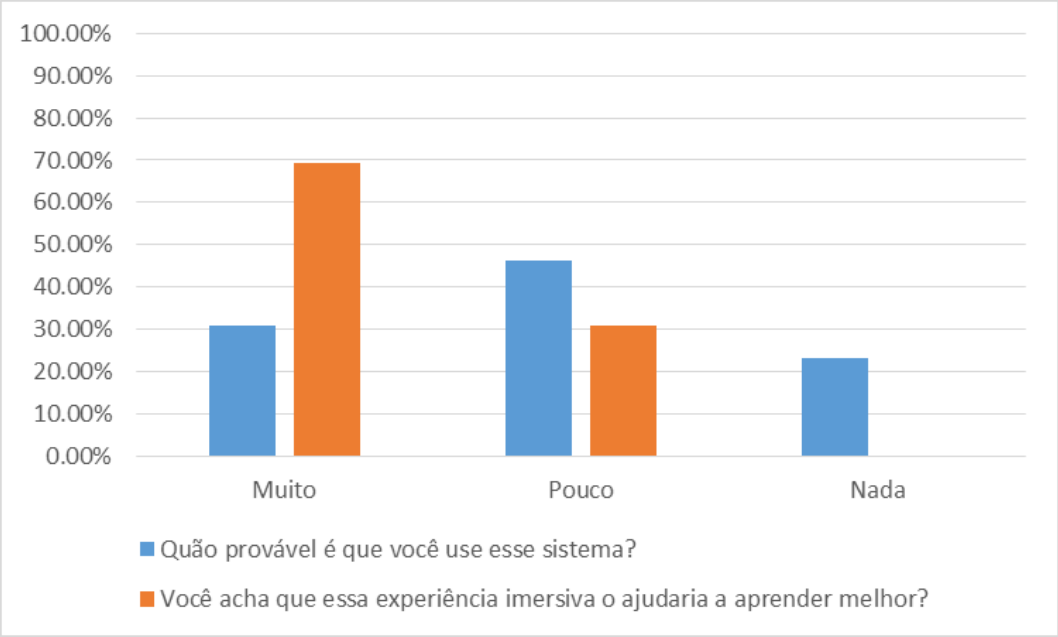


Figura 6.14 Histograma que mostra o resultado dos participantes respeito à usabilidade da ferramenta

Pode-se notar que os participantes em sua maioria indicam que a ferramenta virtual os ajudaria a aprender melhor, além de que a pergunta sobre se usariam o sistema a maioria indica que muito e pouco. Indiretamente estas perguntas permitiram que os estudantes contribuíssem com algumas ideias de melhoramento para a ferramenta e ensinando com isso seu pensamento crítico frente a la avaliação da ferramenta.

6.3. Observações do comportamento dos participantes dentro da ferramenta virtual.

A seguir se descreve alguns comportamentos que tiveram os participantes durante a imersão no mundo virtual. Pode-se ver quando o carro começou-se a mexer eles apoiaram-se as mãos sobre suas pernas e alguns deles exclamaram "wouuu ...calma", isto pode-se ver na Figura 6.15, na figura esquerda temos uma participante do IRS e na direita um participante da escola Darcy Vargas, ambos tem as mãos em seu joelhos, isto para segurar-se quando o carro começou a avançar.

Figura 6.15 Participantes do IRS (esquerda) e Darcy Vargas (direita) com as mãos nas pernas.

Na seguinte Figura 6.16 pode se ver quando os participantes começaram a mexer os braços quando eles chegaram ao ponto final, isto deve-se a que neste ponto final estão as outras pessoas que formam a audiência. A audiência está fazendo sinais de motivação com seus braços e apoiando a participante para que ele chegue ao ponto final, alguns dos participantes tinham a curiosidade de que o personagem da audiência reagisse frente a um contato com os braços de seu avatar

Figura 6.16 Participantes dentro do ambiente virtual interagindo com os outros personagens

6.4. Discussão

A ferramenta de programação utilizou a técnica do *drag and drop de blocos* própria dos linguagens visuais de programação, esta técnica é muito intuitiva na elaboração do código visual, usou também a seleção do personagem tendo 2 avatares: um avatar que representa os homens e outro as mulheres. e também se usou as cores para diferenciar cada bloco, dependendo se ele é uma função ou uma estrutura de controle. Além disso, há o conjunto de desafios que permite que o usuário use para cada nível uma determinada estrutura de programação (repetitivas e condicional).

A tecnologia de Realidade Virtual ajudou muito a que os estudantes ficassem apegados à ferramenta e conseguissem resolver todos os desafios.

Os participantes gostaram da ferramenta, pois a fácil geração de código através do montagem de blocos e além a imersão no mundo virtual permitiu que eles compararem esta sensação com os jogos tradicionais no computador. Alguns deles tinham o interesse em usar a ferramenta para ensinar a seus colegas de outras turmas e a seus familiares.

Eles também forneceram algumas sugestões de melhora para a ferramenta, como por exemplo propuseram um novo cenário onde o objeto virtual (carro) podia se mover, além de sugerir adicionar seus próprios desafios colocando mais obstáculos. Estas acções de proponer melhorias e novas funcionalidades provocou que os participantes pensassem como cientistas da computação. A maioria dos participantes consideram que essa experiência imersiva os ajudaria a aprender melhor. Um participante comentou que queria levar a tecnologia de realidade virtual a outras matérias como biologia, anatomia e geografia.

Através das respostas da entrevista pré-teste e pós-teste e da análise do comportamento, descobrimos que a experiência de imersão no mundo virtual motivou que eles aprendessem um pouco da programação e sua introdução a ciências da computação. A análise também destacou a necessidade de gerar mais aplicações reais em outras matérias, isto é, que incluíam a experiência de imersão.

7
Conclusões e trabalhos futuros

Este trabalho apresentou o desenvolvimento e a avaliação de uma ferramenta de programação para introduzir os estudantes na aprendizagem do pensamento computacional e da programação, isto por meio da elaboração de algoritmos para dar solução ao conjunto de 10 desafios propostos; para isto os alunos tiveram que trabalhar em grupos de dois alunos por computador. Assim, o presente trabalho objetivou verificar se a imersão dentro do mundo virtual ajudava a que os estudantes se motivassem e conseguissem terminar todos os desafios, e também se gostaram da tecnologia. Dessa forma, o uso dos desafios que foram representados por meio de jogos poderá posteriormente ser utilizado como um apoio para introduzir e motivar os estudantes a conhecer e gostar da programação.

O fato de desenvolver a ferramenta como um compilador visual permitiu ao participante a elaboração do código fonte de uma forma fácil. Isto devido à técnica usada "Drag and drop", que consiste na montagem de blocos, de forma a mitigar os erros de sintaxe e permitir que os participantes concentrem-se mais na lógica de programação. Além disso a ferramenta possui a funcionalidade de apresentar para o participante o código fonte na linguagem de programação C#. Com isto, os alunos sabiam que os blocos eram só representações dos comandos que se usam para programar.

O processo de desenvolvimento consiste em analisar o desafio, propor uma solução, avaliar e depurar a solução, corrigir a proposta caso ela tenha erros e otimizar a solução. Este processo aplicado a cada desafio ajudou muito a que os participantes conhecessem de uma forma indireta os conceitos e comportamentos do pensamento computacional, e como este serviu para gerar a solução.

Observou-se que o conjunto de desafios permitiu que os participantes se sentissem atraídos pela ferramenta, ao mesmo tempo em que gerou competições entre eles, procurando sempre quem terminava antes de todos os desafios. Além disso, o trabalhar em duplas permitiu a comunicação e o intercâmbio de conhecimento; esta ação permitiu a eles conhecer uma forma de trabalho comum

entre os cientistas da computação, conhecida como "Pair-Programing". Alguns dos estudantes também começaram a propor outros tipos de soluções para os desafios; depois de fazer algumas comparações com outros estudantes, eles começaram a entender que para resolver um problema não existe só uma forma de solução. Isto permitiu entender que os desafios têm soluções ótimas e soluções não ótimas, onde os critérios para decidir que solução eleger poderiam ser a quantidade de blocos utilizados ou o tempo que se precisava para chegar ao objetivo.

As funcionalidades que foram implementadas na ferramenta, como por exemplo "Depurar" e "Executar", permitiram aos estudantes encontrar o problema onde o script de código estava mal elaborado. Além disso, a ferramenta ajudou a visualizar o comportamento errado do objeto móvel, isto é, quando ele não conseguia chegar ao objetivo.

O uso da realidade virtual permitiu a conexão entre o participante e a ferramenta, isto notou-se mediante as expressões deles quando viam seu corpo apresentado pelo avatar, que foi um dos principais fatores pelo qual eles ficavam ligados à ferramenta até cumprir com todos os desafios. Na imersão do participante no mundo virtual destacou-se os 3 princípios básicos "Telepresença", "Presença Social" e a "Usabilidade"; a ferramenta de programação atendeu todos estes três princípios, confirmado posteriormente pelas opiniões no questionário do pós-teste.

O uso das avaliações divididas em pré-teste, teste e pós-teste permitiu conhecer o estado inicial dos participantes antes de começar com o teste. As perguntas formuladas, como por exemplo "O que é um algoritmo?", "O que é programar?" ou “conhecer se eles tiverem alguma experiência usando algum aparelho de realidade virtual” deu um panorama para poder conversar com eles e explicar os objetivos do estudo e posteriormente cumprir um dos objetivos da pesquisa, que era ensinar a eles de uma forma introdutória a programação. O pós-teste permitiu avaliar todo o aprendido, neste considerou-se repetir algumas perguntas do pré-teste, isto para conhecer como eles iriam responder as perguntas que no pré-questionário não sabiam. Além disso, este questionário ajudou a ter um feedback por parte deles, como por exemplo algumas sugestões com respeito ao desenho gráfico e os níveis de complexidade dos desafios, e conhecer também as opiniões que trazem sobre a realidade virtual e a programação.

O estudo feito em duas escolas diferentes do Rio de janeiro permitiu conhecer as diferenças destes estudantes, isto é, os participantes da escola Darcy Vargas tiveram algumas dificuldades a mais que os participantes do IRS, isto porque os participantes do IRS são alunos que anteriormente já haviam trabalhado com programação em outras ferramentas visuais. Além disso, a diferença notou-se na quantidade de perguntas que faziam ao guia.

Como trabalhos futuros, existem as modificações acima mencionadas pelos participantes que se fazem necessárias para uma maior adequação da ferramenta aos usuários. Além disso, seria interessante trabalhar com a integração de mais elementos da programação como funções, variáveis, operações matemáticas e entrada de dados, o que levaria à elaboração de desafios mais complexos que aprofundem mais o conhecimento da programação e o pensamento computacional. A implementação de mais componentes à ferramenta permitirá com que os participantes gerem seus próprios desafios.

Fazer com que o participante interaja mais no mundo virtual. Pode-se por exemplo usar uma técnica de gameficação, que consiste em que os participantes durante a imersão possam colecionar objetos com a movimentação de suas extremidades (braços e pernas) e estes objetos apresentem algum bônus, gerando com isto uma competição entre os participantes, o que ajudaria a mantê-los mais ligados e entusiasmados ao uso da ferramenta.

Poderia-se usar a ferramenta para o ensino de alguma outra matéria de ciências, isto é, mediante a elaboração de desafios focados numa matéria específica podendo ser: matemática, física, biologia, entre outras.

Para uma futura avaliação, seria bom levar em consideração uma maior quantidade de participantes divididos por níveis de educação, isto para uma gerar um melhor resultado e conhecer melhor as opiniões deles, o que servirá para a elaboração de desafios adequados para um certo nível de ensino. Além disso, elaborar uma avaliação com estes mesmos participantes usando ferramentas comuns, isto é, que não usem a realidade virtual.

8
Referências bibliográficas

Ahamed, S. I., Brylow, D., Ge, R., Madiraju, P., Merrill, S. J., Struble, C. A., & Early, J. P. (2010). **Computational thinking for the sciences**. In Proceedings of the 41st ACM technical symposium on Computer science education - SIGCSE '10. https://doi.org/10.1145/1734263.1734277

Arraki, K., Blair, K., Burgert, T., Greenling, J., Haebe, J., Lee, G., … Hug, S. (2014). **DISSECT: An experiment in infusing computational thinking in K-12 science curricula.** In 2014 IEEE Frontiers in Education Conference (FIE) Proceedings. https://doi.org/10.1109/fie.2014.7044262

Ater-Kranov, A., Bryant, R., Orr, G., Wallace, S., & Zhang, M. (2010). **Developing a community definition and teaching modules for computational thinking.** In Proceedings of the 2010 ACM conference on Information technology education - SIGITE '10. https://doi.org/10.1145/1867651.1867689

Berland, M., & Lee, V. R. (2011). **Collaborative Strategic Board Games as a Site for Distributed Computational Thinking**. International Journal of Game-Based Learning, 1(2), 65–81.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). **Computational thinking and tinkering: Exploration of an early childhood robotics curriculum.** Computers & Education, 72, 145–157.

Bers, M.U., (2010). **The TangibleK Robotics program: Applied computational thinking for young children.** Early Childhood Research & Practice, 12(2), p.n2.

Blockly | Google Developers. (n.d.). Retrieved October 01, 2017, from https://developers.google.com/blockly/

BLS, Inc. Fastest growing occupations. (n.d.). Retrieved October 9, 2017, from http://www.bls.gov/emp/ep_table_103.htm

Blum, L., & Cortina, T. J. (2007). **CS4HS.** In Proceedinds of the 38th SIGCSE technical symposium on Computer science education - SIGCSE '07. https://doi.org/10.1145/1227310.1227320

Boechler, P., Artym, C., Dejong, E., Carbonaro, M., & Stroulia, E. (2014). **Computational Thinking, Code Complexity, and Prior Experience in a Videogame-Building Assignment.** In 2014 IEEE 14th International Conference on Advanced Learning Technologies. https://doi.org/10.1109/icalt.2014.118

Chang, S.-K. (1990). **A visual language compiler for information retrieval by visual reasoning**. IEEE Transactions on Software Engineering, 16(10), 1136–1149.

CiMPLE, Retrieved January 15, 2017, from http://cimple.sofware.informer.com/.

Daily, S. B., Leonard, A. E., Jörg, S., Babu, S., & Gundersen, K. (2014). **Dancing alice**. In Proceedings of the 45th ACM technical symposium on Computer science education - SIGCSE '14. https://doi.org/10.1145/2538862.2538917

Dasgupta, S., Hale, W., Monroy-Hernández, A., & Hill, B. M. (2016). **Remixing as a Pathway to Computational Thinking**. In Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing - CSCW '16. https://doi.org/10.1145/2818048.2819984

Denning, P. J. (2009). **The profession of ITBeyond computational thinking**. Communications of the ACM, 52(6), 28.

Denny, P., Luxton-Reilly, A., Tempero, E., & Hendrickx, J. (2011). **Understanding the syntax barrier for novices.** In Proceedings of the 16th annual joint conference on Innovation and technology in computer science education - ITiCSE '11. https://doi.org/10.1145/1999747.1999807

Dwyer, H. A., Boe, B., Hill, C., Franklin, D., & Harlow, D. (2014). **Computational Thinking for Physics: Programming Models of Physics Phenomenon in Elementary School.** In 2013 Physics Education Research Conference Proceedings. https://doi.org/10.1119/perc.2013.pr.021

Freudenthal, E., Ogrey, A. N., Roy, M. K., & Siegel, A. (2010). **A computational introduction to STEM studies.** In IEEE EDUCON 2010 Conference. https://doi.org/10.1109/educon.2010.5492514

Fronza, I., El Ioini, N., & Corral, L. (2015). **Students Want to Create Apps.** In Proceedings of the 16th Annual Conference on Information Technology Education - SIGITE '15. https://doi.org/10.1145/2808006.2808033

Game Engine Technology by Unreal. (n.d.). Retrieved October 9, 2017, from https://www.unrealengine.com/

Goldberg, D. S., Grunwald, D., Lewis, C., Feld, J. A., & Hug, S. (2012). **Engaging computer science in traditional education.** In Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education - ITiCSE ’12. https://doi.org/10.1145/2325296.2325377

Golin, E. J., & Reiss, S. P. (1990). **The specification of visual language syntax. Journal of Visual Languages & Computing,** 1(2), 141–157.

Grover, S., Cooper, S., & Pea, R. (2014). **Assessing computational learning in K-12.** In Proceedings of the 2014 conference on Innovation & technology in computer science education - ITiCSE ’14. https://doi.org/10.1145/2591708.2591713

Grover, S., Pea, R., & Cooper, S. (2016). **Factors Influencing Computer Science Learning in Middle School.** In Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE ’16. https://doi.org/10.1145/2839509.2844564

Guzdial, M. (2008). **Education Paving the way for computational thinking.** Communications of the ACM, 51(8), 25.

Guzdial, M., (2011). A**ny cognitive benefit of video games? Video-game studies have serious flaws.** Retrieved 10 December 2011, from

Haden, P and Mann, S. (2003): **The Trouble With Teaching Programming**, Proceedings of the NACCQ, Palmerston North, New Zealand, 63-70.

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). **A multidisciplinary approach towards computational thinking for science majors.** ACM SIGCSE Bulletin, 41(1), 183.

Imberman, S.P., Sturm, D. and Azhar, M.Q., 2014. **Computational thinking: expanding the toolkit.** Journal of Computing Sciences in Colleges, 29(6), pp.39-46.

Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2011). **Understanding Computational Thinking before Programming.** International Journal of Game-Based Learning, 1(3), 30–52.

Kelleher, C., Pausch, R., & Kiesler, S. (2007). **Storytelling alice motivates middle school girls to learn computer programming.** In Proceedings of the SIGCHI conference on Human factors in computing systems - CHI ’07. https://doi.org/10.1145/1240624.1240844

Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). **A study of the difficulties of novice programmers.** In Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education - ITiCSE '05. https://doi.org/10.1145/1067445.1067453

Levy, P., & da Costa, C. I. (1993). **tecnologias da inteligência**, As. Editora 34.

Microsoft Kinect - Windows app development. (n.d.). Retrieved October 9, 2017, from https://www.microsoft.com/en-us/kinectforwindows/meetkinect/default.aspx

Nesiba, N., Pontelli, E., & Staley, T. (2015). DISSECT: **Exploring the relationship between computational thinking and English literature in K-12 curricula.** In 2015 IEEE Frontiers in Education Conference (FIE). https://doi.org/10.1109/fie.2015.7344063

Oculus Rift Development Kit 2 (DK2) | Oculus. (n.d.). Retrieved October 9, 2017, from https://www.oculus.com/en-us/dk2/

Parmar, D., Isaac, J., Babu, S. V., D'Souza, N., Leonard, A. E., Jorg, S., … Daily, S. B. (2016). **Programming moves: Design and evaluation of applying embodied interaction in virtual environments to enhance computational thinking in middle school students.** In 2016 IEEE Virtual Reality (VR). https://doi.org/10.1109/vr.2016.7504696

Pierce, J. S., Shochet, J., Staack, D., Stearns, B., Sturgill, C., Williams, G., … Patten, J. (1997). **Alice: Easy to Use Interactive 3D Graphics.** In Proceedings of the 10th annual ACM symposium on User interface software and technology - UIST '97. https://doi.org/10.1145/263407.263512

Pollock, L., Mouza, C., Atlas, J., & Harvey, T. (2015). **Field Experiences in Teaching Computer Science.** In Proceedings of the 46th ACM Technical Symposium on Computer Science Education - SIGCSE '15. https://doi.org/10.1145/2676723.2677286

Pure Data — Pd Community Site. (n.d.). Retrieved October 9, 2017, from https://puredata.info/

Qualls, J. A., & Sherrell, L. B. (2010). **Why computational thinking should be integrated into the curriculum.** J. Comput. Small Coll., 25(5), 66-71

Repenning, A., Webb, D., & Ioannidou, A. (2010). **Scalable game design and the development of a checklist for getting computational thinking into public**

schools. In Proceedings of the 41st ACM technical symposium on Computer science education - SIGCSE ’10. https://doi.org/10.1145/1734263.1734357

Robertson, S., & Robertson, J. C. (2006). **Mastering the Requirements Process.** Pearson Education.

Robins, A., Rountree, J., & Rountree, N. (2003). **Learning and Teaching Programming: A Review and Discussion**. Computer Science Education, 13(2), 137–172.

Rubinstein, A., & Chor, B. (2014). **Computational thinking in life science education.** PLoS Computational Biology, 10(11), e1003897.

Scratch - Imagine, Program, Share. (n.d.). Retrieved October 9, 2017, from https://scratch.mit.edu/

Shailaja, J. and Sridaran, R., (2015). **Computational Thinking the Intellectual Thinking for the 21st century**. International Journal of Advanced Networking & Applications, May 2015 Special Issue, pp.39-46.

Sherman, W. R., & Craig, A. B. (2003). **Understanding Virtual Reality—Interface, Application, and Design. Presence:** Teleoperators and Virtual Environments, 12(4), 441–442.

StarUML. (n.d.). Retrieved October 9, 2017, from http://staruml.io/

Sysło, M.M. and Kwiatkowska, A.B., 2014 **Learning Mathematics supported by computational thinking. Constructionism and Creativity**, pp.258-268

Van den Branden, K., Van Gorp, K., & Verhelst, M. (2009). **Tasks in Action: Task-Based Language Education from a Classroom-Based Perspective.** Cambridge Scholars Publishing.

Van Dyne, M., & Braun, J. (2014). **Effectiveness of a computational thinking (CS0) course on student analytical skills**. In Proceedings of the 45th ACM technical symposium on Computer science education - SIGCSE ’14. https://doi.org/10.1145/2538862.2538956

VisSim Products. (n.d.). Retrieved October 9, 2017, from http://vision-traffic.ptvgroup.com/en-us/products/

Whimbey, A., Lochhead, J., & Narode, R. (2013). **Problem Solving & Comprehension. Routledge**.

Wing, J. M. (2006). **Computational thinking**. Communications of the ACM, 49(3), 33.

Wing, J.M., (2008). **Computational thinking and thinking about computing.** Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences, 366(1881), pp.3717-3725.

Apêndice A
Especificação de Casos de Uso de sistema

Nome de Caso de Uso: Criar novo documento.

Nome de Caso de Uso	Criar novo documento.
Ator Principal	Usuário
Atores Secundários	--
Resumo	Neste caso de uso o usuário poderá eleger a opção de criar documento novo, e o sistema deverá apresentar uma janela de cor branca sem nenhum bloco.
Pré-Condição	Nenhum
Pós-Condição	Se apresenta um quadro cor branco no lado direito da ferramenta.
Fluxo Básico	1. O sistema apresenta a opção "criar novo documento" 2. O usuário elege a opção "Criar novo documento" O sistema verifica que o documento foi salvo. 3. Em caso do documento já ter sido salvo, o sistema apaga todos os objetos e blocos da janela de "Edição de código" 4. O sistema deixa em branco a janela de "Edição de código".
Fluxo Alternativo	No caso que o usuário não salvou seu trabalho no passo (3) 1. O sistema mostra a seguinte mensagem de alerta "Deseja salvar o script" com as opções "Sim" e "Não". 2. O usuário elege a opção "Sim", então o sistema salva o documento e volta para o passo (3). 3. No caso que o usuário elege "Não" o sistema apaga todo o conteúdo da janela e vai para o passo (4).

Nome de Caso de Uso: Mexer Bloco

Nome de Caso de Uso	Mexer Bloco
Ator Principal	Estudante
Atores Secundários	--
Resumo	Este caso de uso consiste em que o usuário estudante possa arrastar um bloco e largá-lo na janela de edição de código.
Pré-Condição	O usuário está no sistema com o papel de estudante
Pós-Condição	O bloco estará posicionado na posição exata onde o usuário o soltou.
Fluxo Básico	1. O estudante pega um bloco do conjunto de blocos,

	isto na parte esquerda da janela principal. 2. O sistema deverá responder a este evento e apresentará o bloco ressaltado sobre os demais blocos. 3. O usuário ao fazer pressão com o mouse sobre o bloco. 4. O sistema deverá apresentar o bloco mexendo-se, sempre seguindo a direção do mouse.
Fluxo Alternativo	Não aplica

Caso de uso: Unir automaticamente os Bloco

Nome de Caso de Uso	Unir automaticamente os Blocos
Ator Principal	Usuário
Atores Secundários	Sistema
Resumo	Este caso de uso consiste em que o usuário selecione um bloco e o coloque acima do outro bloco que esteja localizado na janela de programação. Posteriormente o sistema colocará o bloco que está sobreposto na parte de baixo do primeiro bloco. Com isto mostra-se a sensação de união dos blocos, levando em conta que a conexão de blocos dá-se pela interseção do "female conector" e "male connector".
Pré-Condição	O usuário traz um bloco à janela de programação.
Pós-Condição	O bloco estará posicionado na janela de edição, na posição correta: no meio dos blocos ou ao final.
Fluxo Básico	O usuário solta o bloco na janela de programação, isto acima da estrutura já formada por outros blocos. O sistema deverá encaixar o bloco dentro da estrutura já formada.
Fluxo Alternativo	No caso 2 se o bloco novo não encaixa na estrutura já formada. 1. O sistema posiciona ao novo bloco fora da estrutura já formada.

Caso de uso: Separar automaticamente os Blocos.

Nome de Caso de Uso	Separar automaticamente os Blocos
Ator Principal	Usuário
Atores Secundários	Sistema
Resumo	Este caso de uso consiste em que o sistema separe os blocos que estavam unidos, isto depois de que o usuário pegue o bloco e o mexa com ele 3px de separação.
Pré-Condição	O usuário traz um bloco à janela de programação.
Pós-Condição	O bloco estará separado do bloco base, o bloco base estará a uma distância de 3px.
Fluxo Básico	1. O usuário seleciona um bloco da janela de programação, este bloco está conectado com outros

	blocos. 2. O usuário mexe o bloco 3px de distância do bloco base. 3. O sistema automaticamente libera o bloco selecionado do bloco base. 4. O usuário solta o bloco selecionado. 5. O sistema deverá colocar o bloco que foi solto na nova ubicação, isto é, 3px longe do bloco base.
Fluxo Alternativo	No caso 2 se o bloco selecionado não é mexido uma distância maior que 3 px. 1. O sistema não realiza nenhuma ação, isto é, o bloco permanece no mesmo lugar.

Caso de Uso: Aumentar o tamanho vertical dos Blocos

Nome de Caso de Uso	Aumentar o tamanho vertical do bloco composto
Ator Principal	Usuário
Atores Secundários	Sistema
Resumo	Este caso de uso consiste em que o sistema mude o tamanho do bloco composto, isto é, quando o usuário insere um novo bloco dentro dele.
Pré-Condição	O usuário deverá soltar um bloco dentro do bloco composto na janela de programação.
Pós-Condição	O bloco composto muda de tamanho em forma vertical
Fluxo Básico	1. O estudante solta o bloco na janela de programação, isto acima do bloco composto. 2. O sistema automaticamente muda o tamanho do bloco composto, isto com intenção de encapsular o novo bloco que foi inserido.
Fluxo Alternativo	--

Caso de Uso: Diminuir o tamanho vertical dos Blocos

Nome de Caso de Uso	Diminuir o tamanho vertical dos Blocos
Ator Principal	Usuário
Atores Secundários	Sistema
Resumo	Este caso de uso consiste em que o sistema diminui o tamanho do bloco composto, isto acontece quando o usuário tira um bloco que está dentro dele.
Pré-Condição	O usuário tira um bloco que está dentro do bloco composto.
Pós-Condição	O bloco se mostra com o tamanho reduzido.
Fluxo Básico	1. O estudante pega um bloco que está dentro de outro bloco (composto) 1. O sistema automaticamente atualiza o tamanho do bloco composto, isto é, reduz o tamanho na mesma quantidade do bloco que foi tirado.
Fluxo Alternativo	--

Apêndice B
Questionário pré-teste

I. DADOS PESSOAIS

- Nome: .....
- Idade: .....
- Sexo: Masculino ( ) ou Feminino ( )
- Ano de estudo: .....

II. EXPERIÊNCIA NO USO DE DISPOSITIVOS ELETRÔNICOS

- **Com que dispositivo (s) você interage? (Pode marcar mais de um)**

  ( ) Caixas eletrônicos

  ( ) Computador Desktop

  ( ) Computador Notebooks

  ( ) Smartphone

  ( ) Tablet

  ( ) Outro(s) especifique:.....

  .....

*****Responder a seguintes perguntas em caso de haver selecionado o computador (Desktop/Notebooks)

- **Frequência de utilização do computador:**

  ( ) Pelo menos uma vez por dia

  ( ) Pelo menos uma vez por semana

  ( ) Raramente

  ( ) Nunca

- **Com que fim usa o computador:**

  ( ) Jogos e entretenimento

  ( ) Internet

  ( ) Aplicativos e ferramentas de desenho (Ex.: Paint)

  ( ) Outro (s) especifique: .....

  .....

- **O que dispositivos de Realidade Virtual você utilizo?**

( ) Oculus Rift

( ) HTC Vive

( ) Google Cardboard

( ) Google Gear VR

( ) PlayStation VR

*Responder a seguintes perguntas em caso de haver utilizado algum dispositivo de realidade virtual.

- **Qual (s) tipo (s) de aplicação 3D você utiliza?**

( ) Modelagem 3D

( ) Visualização 3D

( ) Jogos 3D

( ) Simuladores

( ) Outro(s): .....

.....

- **Qual sua experiência utilizando aplicações de realidade virtual?**

( ) Profunda

( ) Boa

( ) Média

( ) Pouca

( ) Nenhuma

III. CONHECIMENTO DA CIÊNCIA DA COMPUTAÇÃO

- **O que é um algoritmo?**

( ) é uma receita de cozinha

( ) um conjunto de passos ordenados para resolver algo

( ) é uma parte do computador

( ) não tenho ideia

- **O que é programar?**

( ) é usar as aplicações do computador word, facebook, twitter, email, etc.

( ) é reparar o computador

( ) é escrever instruções para que o computador o faça

( ) não tenho ideia

- **Utilizo alguns destes softwares? (Pode marcar várias opções)**

( ) Scratch

( ) App Inventor

( ) Alice

( ) C++ ( ) Java ( ) Python

( ) Unity

- **O que os cientistas da computação fazem?**

( ) Não tenho ideia

( ) Criam Software

( ) Fazem/Consertam/Criam Computadores

( ) Escrevem Código/Programas

( ) Fazem filmes/Videogames/Animações

( ) Outro (s) .....

.....

Apêndice C
Questionário pós-teste

A. Entrevista para o participante

Por favor, responda o questionário a seguir sobre a sua interação no último cenário.

Nas questões marque um X de acordo com a escala que representa o seu grau de concordância com a afirmação.

	Discordo totalmente				Concordo totalmente
Foi fácil mexer os blocos e encontrar os blocos que você procurava.	1	2	3	4	5
Foi fácil gerar o Script de código para enfrentar o (1) nível.	1	2	3	4	5
Foi fácil gerar o Script de código para enfrentar o (2) nível.	1	2	3	4	5
Foi fácil gerar o Script de código para enfrentar o (3) nível.	1	2	3	4	5
Foi fácil gerar o Script de código para enfrentar o (4) nível.	1	2	3	4	5
Foi fácil gerar o Script de código para enfrentar o (5) nível.	1	2	3	4	5
Foi fácil gerar o Script de código para enfrentar o (6) nível.	1	2	3	4	5
Foi fácil gerar o Script de código para enfrentar o (7) nível.	1	2	3	4	5
Foi fácil gerar o Script de código para enfrentar o (8) nível.	1	2	3	4	5
Foi fácil gerar o Script de código para enfrentar o (9) nível.	1	2	3	4	5
Foi fácil gerar o Script de código para enfrentar o (10) nível.	1	2	3	4	5

Marque os 3 níveis mais difíceis que você afrontou?

1	2	3	4	5	6	7	8	9	10

Conhecimento da Ciência da Computação

- **O que os cientistas da computação fazem?**

( ) Criam Software

( ) Fazem/Consertam/Criam Computadores

( ) Escrevem Código/Programas

( ) Fazem filmes/Videogames/Animações

( ) Outros

- **O que é um algoritmo?**

( ) é uma receita de cozinha

( ) um conjunto de passos ordenados para resolver algo

( ) é uma parte do computador

( ) não tenho ideia

- **O que é programar?**

( ) é usar as aplicações do computador word, facebook, twitter, email, etc.

( ) é reparar o computador

( ) é escrever instruções para que o computador o faça

( ) não tenho ideia

Avaliação da interação imersiva na ferramenta

Tele presença

- Você sentiu que estava dentro e cercado pelo meio ambiente?
- Como se sentiu ao ver seu próprio corpo lá?
- Você sentiu que estava no labirinto acima do carro?

Social Presença

- O personagem se sentiu real?
- Como sentiu-se quando era incentivado pelo público?
- A pessoas que aprecem na ferramenta parecem reais?

Usabilidade, satisfação e entusiasmo

- Quão provável é que você use esse sistema?
- O que você mudaria nesse sistema?
- Você acha que essa experiência imersiva o ajudaria a aprender melhor?

B. Entrevista para o pesquisador

- De uma forma geral, o que o participante achou do uso da ferramenta?
- Pedir para o participante esclarecer pontuações negativas do questionário
- Perguntar ao participante se ele possui sugestões de melhorias

C. Entrevista para o professor

Prezado (a) professor: Solicito sua colaboração para responder às seguintes perguntas com informações sobre sua opinião a respeito dos resultados dos testes.

- De uma forma geral, o que você achou do desempenho dos participantes no teste?
- Como você acha que foi o desempenho no primeiro nível para cada participante?
- Como foi o primeiro contato dele com a tecnologia de realidade Virtual?
- Houve dificuldades no entendimento da atividade?
- Você acha que de alguma forma o uso das novas tecnologias de Realidade Virtual, promoveu o interesse deles em aprender a programar?

Apêndice D
Termo de consentimento livre e esclarecido

I. NATUREZA DA PESQUISA

Você e seu(sua) filho(a) estão sendo convidados a participar de uma pesquisa sobre a utilização de uma ferramenta que ajuda as crianças a programar intitulada "Ferramenta de programação visual para a introdução ao pensamento computacional e programação fazendo uso da Realidade Virtual", sob orientação do Professor Dr. Alberto Barbosa Raposo, do Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) e com o apoio da Sr/Sra. .....

A pesquisa tem como finalidade investigar o impacto da realidade virtual no melhoramento da atenção e motivação para a aprendizagem da programação de computadores. Seu filho(a) poderá ser filmado(a) e fotografado(a) nas sessões semanais na qual participaram com todos seus companheiros de aula, o tempo de avaliação serão de 2 horas durante 3 semanas que começaram o 11 de agosto do 2017.

II. DESCONFORTOS, RISCOS E BENEFÍCIOS

A participação neste estudo não traz complicações, riscos ou desconforto para os participantes. Os procedimentos utilizados nesta pesquisa seguem as normas estabelecidas pelo Estatuto da Criança e do Adolescente (LEI No. 8.069, de 13/07/1990) e não oferecem nenhum risco à integridade física, psíquica e moral de seu(sua) filho(a). Ao participar desta pesquisa você e seu(sua) filho(a) não deverão ter nenhum benefício direto. Entretanto, esperamos que esta pesquisa informe-nos sobre o impacto que gera a realidade virtual no aprendizagem da programação nos estudantes de ensino fundamental. No futuro, essas informações poderão ser utilizadas em benefício de usuários brasileiros que desejem aprofundar mais no estudo da Realidade Virtual com enfoque na educação.

III. FORMA DE ACOMPANHAMENTO. ASSISTÊNCIA E ESCLARECIMENTOS

Você poderá receber cópias dos relatórios da pesquisa contendo os resultados do estudo. Sempre que julgar necessário você poderá solicitar mais informações sobre a pesquisa entrando em contato com o pesquisador pelo telefone (21) 980506455 ou pelo e-mail hcurasma@inf.puc-rio.br.

IV. LIBERDADE DE RECUSA E RETIRADA DE CONSENTIMENTO

Vocês foram selecionados por apresentarem os requisitos básicos para o procedimento a ser utilizado na pesquisa e sua participação não é obrigatória. A qualquer momento você e seu filho(a) podem desistir de participar e retirar seu consentimento. Sua recusa não trará nenhum prejuízo em sua relação com o pesquisador e com a instituição.

V. GARANTIA DE SIGILO

Todas as informações coletadas neste estudo são estritamente confidenciais. Apenas os pesquisadores terão conhecimento dos dados. Se você der a sua autorização por escrito, assinando a Permissão para Utilização de Imagens em Vídeo, os dados poderão ser utilizados para fins de ensino e durante encontros e debates científicos.

Eu (nome do pai ou tutor), ..., abaixo assinado, declaro que:

1. Recebi informações detalhadas sobre a natureza e objetivos do estudo acima, destinado a investigar o impacto do uso da Realidade Virtual para motivar aos estudantes na aprendizagem da programação, sendo que a minha participação e de meu filho não implicará nenhum ônus;
2. Autorizo voluntariamente a participação de meu filho(a) no estudo acima: a) oferecendo informações por meio de entrevistas se necessário e b) autorizando o uso destas informações para finalidades científicas e acadêmicas, desde que garantido sigilo sobre minha identidade e a identidade do(a) meu(minha) filho(a);
3. Tenho conhecimento de que sou livre para desistir de participar do estudo a qualquer momento, com garantias de não ocorrência de constrangimentos ou represálias, sem necessidade de justificar minha decisão e, neste caso, comprometo-me a avisar o pesquisador;
4. Tenho conhecimento de que minha participação é sigilosa, isto é, que minha identidade não será divulgada em qualquer publicação, relatório ou comunicação científica referentes aos resultados da pesquisa;

5. Estou de acordo que as atividades previstas no estudo não representam nenhum risco para mim e meu filho(a) ou para qualquer outro participante,

Nome da criança:

....., de ..... de 2017.

_____

Assinatura do responsável legal (Pai ou tutor)

_____

Assinatura do pesquisador

Herminio Paucar Curasma