INTERFACE DE USUÁRIO TANGÍVEL PARA TRABALHAR COM O PENSAMENTO COMPUTACIONAL NO FURBOT

Jonathan Michels Kuntz, Dalton Solano dos Reis – Orientador

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

jonathan.michels7@gmail.com, dalton@furb.br

Resumo: *Este artigo apresenta o processo de desenvolvimento de um módulo que tem como objetivo auxiliar o aprendizado computacional com interface tangível no FURBOT, assim proporcionando a imersão do usuário na resolução dos exercícios. A interface foi desenvolvida utilizando a plataforma Unity com scripts em C#. Para realizar o processamento de imagem das peças, fazendo assim a interação do mundo real com o mundo digital, foi utilizado a biblioteca OpenCV. A aplicação está disponível para plataforma Android. Foi alcançado o resultado esperado, conseguindo reconhecer as peças posicionadas no mundo real para o mundo digital. Sendo assim é possível utilizar a mesma nos ambientes escolares, mas controlando as limitações, prevenindo assim o mal funcionamento da aplicação.*

Palavras-chave*: Interface de usuário tangível. Unity. OpenCV. Processamento de Imagem. FURBOT.*

1 INTRODUÇÃO

Os últimos anos tem sido marcado no Brasil e no mundo por mudanças educacionais em que a predominância do uso de novas tecnologias tem se destacado (DINIZ, 2001). Para Koch (2013, p.11), "a educação se depara com um duplo desafio: adaptar-se aos avanços das tecnologias e orientar o caminho de todos para o domínio e a apropriação crítica desses novos meios". Segundo Thoaldo (2010, p.9), "a educação no mundo de hoje tende a ser tecnológica, por isso, exige entendimento e interpretação, tanto dos professores quanto dos alunos em relação a essas novas tecnologias".

Conforme Tolentino (2013, p.20), "os alunos dentro do contexto escolar utilizam essas tecnologias o tempo todo, na hora da entrada, no intervalo na hora da saída e possível observar os alunos com celulares, tablet e computadores portáteis". Segundo Otto (2016, p.6), "apesar de todas as vantagens oferecidas, deve-se também analisar a forma que as tecnologias nas escolas devem ser introduzidas e os limites que devem ser respeitados". Alguns professores ainda acabam tendo receio quando se fala de introduzir as tecnologias no ambiente escolar.

Um dos benefícios que a introdução da tecnologia na escola poderia trazer é trabalhar com o pensamento computacional. Ignácio (2018) afirma que o pensamento computacional que pode fazer com que as pessoas, ao atuarem nas diversas áreas, tenham maior facilidade para organizar o pensamento, para assim resolver problemas e trabalhar de forma colaborativa.

> O ganho cognitivo da inserção do pensamento computacional na educação básica está em empoderar jovens estudantes na forma de proceder à resolução de problemas, em sua capacidade para descrever e explicar situações complexas. Estudantes investidos do poder de uma ferramenta cognitiva para resolver problemas de forma mais ágil e apoiados na transversalidade das diferentes áreas do conhecimento passam a analisar dados logicamente e a representá-los de forma abstrata; a espacializar as etapas do processo de resolução de problemas; a particionar problemas complexos, resolvendo-os por meio da discussão de variáveis e de estruturas condicionais. (CONFORTO et al., 2018, p. 103).

Diante do assunto abordado, este trabalho tem como objetivo criar um módulo para utilização de interface de usuário tangível no FURBOT, assim proporcionando a imersão do usuário na resolução dos exercícios. Os objetivos específicos são: disponibilizar a programação dos movimentos do robô com interface de usuário tangível; disponibilizar interface 2D com a programação informada pelo usuário; disponibilizar a simulação dos movimentos do robô; criar as peças de ações para utilizar na aplicação.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os aspectos teórico para o desenvolvimento desta aplicação. Na primeira seção se aborda brevemente os conceitos base para o desenvolvimento da aplicação, de Interface de Usuário Tangível e Pensamento Computacional. Na segunda seção apresenta a versão anterior do FURBOT. Na terceira seção são apresentados três trabalhos correlatos a este artigo.

2.1 INTERFACES DE USUÁRIO TANGÍVEL

Interface de Usuário Tangível (Tangible User Interfaces - TUI) é a interação com objetos virtuais que ocorre utilizando objetos localizados no "mundo real", podendo ser meios de entrada ou saída (JORGE, 2012). Esses objetos

do mundo real podem ser um cubo, bola, copo ou qualquer objeto do uso do dia a dia. Jorge (2012, p.21) diz que, "quando qualquer objeto sendo rastreado pelo sistema é manipulado por usuário, a alteração sofrida pelo objeto é refletida e interpretada pelo sistema e poderá ser gerada uma mudança no estado e na saída de dados do sistema". As TUIs seguem as seguintes premissas (SANT' ANNA; FERRONATO, 2017):

a) tenha algum evento realizado pelo usuário, normalmente algum movimento como pressionar, sacudir, empurrar, girar e mover um objeto fisicamente;

b) a aplicação detecta esse evento e faz o processamento, alterando assim o seu estado interno;

c) por fim, a aplicação executa um evento de saída, a partir da mudança física.

Os objetos das aplicações que utilizam Interface de Usuário Tangível, acabam tendo grandes restrições físicas e não digitais como as aplicações que utilizam outros tipos de interface, pois os objetos do mundo real não conseguem estar no mesmo espaço tempo que outro objeto (JORGE, 2012).

As interfaces tangíveis possuem duas classes definidas por dois parâmetros, sendo eles metáfora e personificação. A metáfora é a que explora a relação dos objetos do mundo real com os digitais, buscando deixar a interação mais natural possível. E a classe de personificação estuda a distância entre as entradas e saídas produzidas, quanto ao dispositivo que capta a entrada e exibe a saída (ANNA; FERRONATO, 2017).

A classe metáfora pode ser dividida entre subcategorias como (ANNA; FERRONATO, 2017):

a) metáfora de nome: O objeto real assemelha-se com o objeto digital, tendo características semelhantes como aparência e som, mas as ações provocadas sobre tal objeto, são diferentes nas provocadas no objeto digital;

b) metáfora de verbo: As aparências entre os objetos real e objeto digital são irrelevantes, mas já as ações, são refletidas entre eles;

c) metáfora completa: Não existe analogia entre os objetos, aqui o objeto do mundo digital é o próprio objeto real.

Personificação pode ser dividida entre subcategorias como (ANNA; FERRONATO, 2017):

a) personificação completa: A interface de entrada e saídas são iguais, assim o dispositivo que captou a entrada exibe a saída;

b) personificação próxima: As interfaces de entrada e saída são próximas, mas mantêm-se separadas.;

c) personificação ambiente: O ambiente do usuário exibe as saídas, em formas de sons, luzes, entre outras;

d) personificação distante: A interface de saída é diferente da que faz o reconhecimento das entradas.

2.2 PENSAMENTO COMPUTACIONAL

Para Wing (2006) Pensamento Computacional ou *computational thinking* não é tentar fazer pessoas pensarem como computadores, mas sim utilizar o raciocínio heurístico na descoberta de uma solução utilizando como base os fundamentos da Ciência da Computação. Wing (2006, p.1) também afirma que, "pensamento computacional é uma habilidade fundamental para todos, não somente para cientistas da computação".

Conforme Costa (2016, p.18), "o pensamento computacional pode ser considerado um pensamento reflexivo que analisa e resolve de forma sucinta os problemas. Ele é um pensamento planejado e estruturado, capaz de identificar quais tarefas cognitivas realizamos melhor do que o computador".

> Portanto, o pensamento computacional é um processo sistematizado, que nos auxilia a compreender desde a concepção do problema até a sua solução, mas para isso, o mesmo se utiliza de um conjunto de conceitos da Ciência da Computação para desenvolver capacidades e habilidades que são primordiais para contribuir durante todo o processo de construção e solução do problema. Tais capacidades e habilidades serão discutidas posteriormente. (COSTA, 2016, p. 20).

Existem quatro pilares que baseiam o Pensamento Computacional (Figura 1), sendo eles (SILVA, 2018):

a) decomposição: que é responsável por quebrar os problemas em pequenas partes, para assim deixar mais fácil encontrar a resolução;

b) reconhecimento de padrões: trabalha com a identificação de similaridades entre os problemas, tornando-se possível replicar as soluções entre os subproblemas;

c) abstração: é deixar somente as informações importantes de uma solução, ignorando aqueles não tão importantes;

d) algoritmo: é o pilar onde agrega todos os outros, sendo assim, é uma sequência de passos para encontrar a melhor solução para um problema.

2.3 VERSÃO ANTERIOR

O Furbot iniciou em 2008, como um *framework* desenvolvido em JAVA. Criado inicialmente para auxiliar nas atividades de ensino da matéria Introdução de Programação, ofertada nos cursos de graduação de Sistema de informação e Ciência da Computação. Posteriormente se passou para formato de jogo, pois percebeu que os jogos digitais acabam muitas vezes sendo mais atrativos, mudando então o proposito, passando agora introduzir o Pensamento Computacional em crianças dos anos iniciais do ensino até adultos. Atualmente o jogo se encontra em desenvolvimento na plataforma Unity utilizando a linguagem C#, com técnicas de *game desing*. É possível encontrar até o momento uma versão beta para dispositivos Android (MATTOS et al, 20192).

O jogo tem como objetivo programar um robô chamado de Furbot, que conta com um ajudante, o drone S-223, que irá auxiliá-lo com dicas para chegar ao fim de cada fase. O robô deve tomar cuidado com obstáculos ao meio do caminho, tentando sempre capturar os objetos no meio da fase que ajudam a terminar a fase, como: energias, tesouros, entre outros. Sempre deve ser tomado cuidado para se manter o robô fora da grama, pois caso o mesmo acabe andando por cima da grama, consome mais energia, fazendo assim acabar a partida por não ter energia suficiente para conclusão da fase (MATTOS et al, 2019).

Após iniciar o jogo e clicar em `Jogar`, é aberto o tutorial obrigatório de visualização do funcionamento. O S-223 aparece indicando e mostrando na tela o funcionamento dos elementos gráficos do jogo. O tutorial foi desenvolvido para simular que adentrou no laboratório da professora Sam e visualiza sua missão (MATTOS et al, 2019).

A programação é feita através do editor localizado no lado direito, quando clicado no mesmo, abre uma aba no lado esquerdo com as possíveis ações a serem escolhidas pelo usuário. De acordo com a seleção de cada comando, o jogo vai indicando qual é a próxima opção válida para seleção. Caso o usuário tenha se precipitado em algum momento, colocando algum comando indesejado, é possível remover o mesmo com a função `Apagar último comando adicionando` ou utilizando a função `Apagar todos os comandos`. Os comandos são executados a partir do momento que o usuário pressionar o botão `executar`. Os comandos do usuário passam pelos analisadores semântico e analítico, se encontrar algum problema é informado para o usuário que existem problemas, se não houver nenhum erro, é executado de forma decrescente, mostrando em negrito cada comando executado (MATTOS et al, 2019). A Figura 1 apresenta como é a estrutura do jogo.

Figura 1 – Tela do jogo

Fonte: elaborado pelo autor.

2.4 TRABALHOS CORRELATOS

Nesta seção são apresentando três trabalhos correlatos, que possuem características relacionadas ao objetivo de estudo deste trabalho. O primeiro trabalho descreve o RoboEduc (CASTRO, 2008), aplicação educacional para ensino da robótica às crianças como uma ferramenta de inclusão digital (Quadro 1). O segundo trabalho descreve o Coding Awbei (OSMO, 2015), aplicação feita para trabalhar com pensamento computacional entre crianças de 5 a 12 anos (Quadro 2). O último trabalho descreve o VisEdu-CG (MONTIBELER, 2014), aplicação didática para visualizar material educacional (Quadro 3).

Quadro 1 – RoboEduc

Referência	Castro (2008).
Objetivos	Facilitar o ensino para pessoas iniciantes no mundo da tecnologia, de forma de desenvolver suas potencialidades.
Principais funcionalidades	Permitir programar de diferentes níveis, desde visual até linguagem textual a ser interpretada ou compilada.
Ferramentas de desenvolvimento	Utilizou a modelagem UML para planejamento da arquitetura da aplicação. Também é utilizado a tecnologia *Lego*, com os kits *Lego Mindstroms*.
Resultados e conclusões	Tanto os alunos que participaram quanto os professores e diretoria da instituição participante, ficaram satisfeitos. O autor conclui falando que a aplicação se mostrou mais indicada à crianças dos oito aos dez anos de idade.

Fonte: elaborado pelo autor.

O trabalho de Castro (2008) possuí níveis diferentes de programação, para que possa ser utilizado com crianças de seis anos até jovens de nível universitário. No primeiro nível a programação é visual, à medida que for passando os níveis irá chegar na linguagem textual a ser interpretada ou compilada. Pode ser utilizada sem conexão com a internet pois funciona off-line.

Quadro 2 – Coding Awbei

Referência	Osmo (2015).
Objetivos	Auxiliar as crianças de 5 a 12 anos a entrar no mundo digital, trabalhando com o pensamento computacional.
Principais funcionalidades	A aplicação identifica os movimentos do robô e explora o ambiente, através da interface tangível.
Ferramentas de desenvolvimento	As ferramentas de desenvolvimento não são acessíveis devido ser uma aplicação comercial.
Resultados e conclusões	O autor diz que teve bons resultados quando aplicou o Conding Awbie em crianças 5 a 9 anos. Quando elas viam o Awbie mastigando os morangos ficavam entretidas. Contudo crianças acima de 9 anos já ficavam entediadas rapidamente com a jogabilidade depois de um tempo.

Fonte: elaborado pelo autor.

O Coding Awbie (OSMO, 2015) é um jogo onde a criança controla um robô chamado de Awbie, utilizando interface de usuário tangível para movimentar as peças e explorar um ambiente. Para utilizar a aplicação é necessário o kit com a base, espelho e as peças para mover o personagem. As peças possuem um imã para facilitar os encaixes e possuem uma sequência lógica para ser encaixada. Cada uma das peças possuem uma cor diferente assim facilitar a sua identificação. Os movimentos possíveis são: andar (azul), pular (vermelho), pegar (laranja), jogar (verde), loop (amarelo) e as direções que cada peça tem (Figura 2).

Figura 2 – Peças utilizadas para movimentar o Awbie

Fonte: Osmo (2019).

Quadro 3 – VisEdu-CG

Referência	Montibeler (2014).
Objetivos	Auxiliar os professores e permitir os alunos praticarem o que foi aprendido em sala de aula.
Principais funcionalidades	A aplicação VisEdu-CG é composta por cinco painéis principais: `Visão da Câmera`: Permite visualizar a cena 3D formada pelo exercício de um ângulo diferente; `Comandos em JOGL`: Mostra códigos fontes em Java que são necessários para reproduzirem o mesmo efeito da peça selecionada; `Fábrica de peças`: Ficam as ações que podem ser executadas chamado de `Fábrica` e o painel de montagem que está denominado como `Renderizador;`
Ferramentas de desenvolvimento	Desenvolvido na linguagem HTML5, utilizando framework Three.js e API WebGL.
Resultados e conclusões	Montibeler (2014) teve resultados positivos, pois mesmo sendo uma aplicação simples, acaba despertando a curiosidade dos alunos, onde começam a "brincar" com a aplicação para ver o que acontece ao ir encaixando as peças nos locais esperados.

Fonte: elaborado pelo autor.

O VisEdu-CG (MONTIBELER, 2014) é uma aplicação web de encaixe de peças com formas geométricas, que permite o estudo de alguns conceitos, sendo eles câmera sintética, grafo de cena, transformações geométricas, composição de transformações geométricas e texturas.

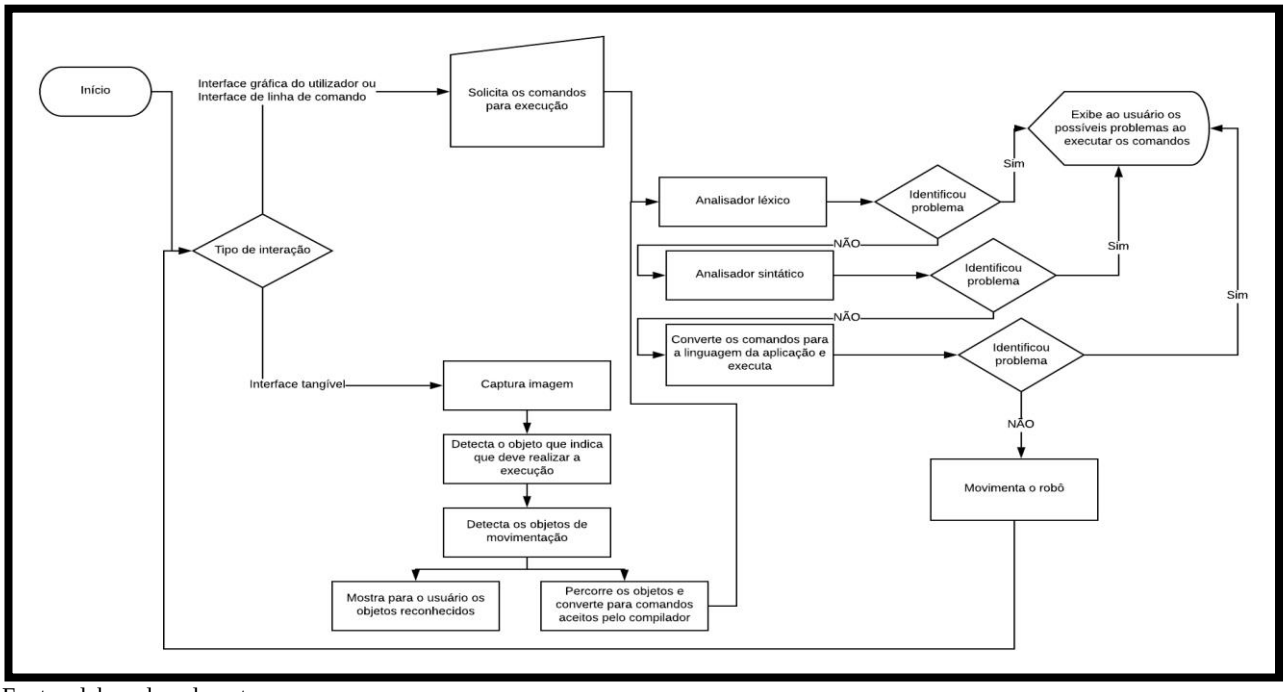
3 DESCRIÇÃO DA APLICAÇÃO

Este capítulo apresenta os detalhes de especificação e implementação da aplicação. Para tanto, são apresentadas três seções. A primeira seção apresenta uma visão geral, descrevendo funcionamento da aplicação e forma de utilização. A segunda seção apresenta as tecnologias utilizadas para a construção da aplicação. A terceira seção apresenta as técnicas utilizadas para o reconhecimento de imagem dos objetos.

3.1 VISÃO GERAL

A aplicação disponibiliza para o usuário uma nova forma de desenvolver o pensamento computacional, onde o usuário movimenta um robô (o robô) em um mundo bidimensional. O processo de movimentação pode ser feito de duas formas. Na primeira opção o usuário tem um editor onde deve ser indicado a sequência de comandos que deseja executar, podendo ser "`andar(DIRECAO);`" (cima, baixo, esquerda e direita) ou laço de repetição tendo algum tipo de critério referente ao cenário, por exemplo o tipo de caminho. A segunda opção detecta os contornos da imagem capturada. E com base nos contornos obtidos é percorrido os mesmos com a intenção de identificar os seguintes objetos: tipos de comandos (laço de repetição ou andar), quantidade de vezes que deve ser repetido e se deve realizar a execução dos objetos encontrados. Após a indicação dos comandos desejados para execução, os mesmos são passados para o compilador. Depois os comandos passam pelos analisadores léxico e sintático e caso os dois retornem verdadeiro os comandos são convertidos e executados na linguagem da aplicação. A Figura 3 exibe o digrama de atividades.

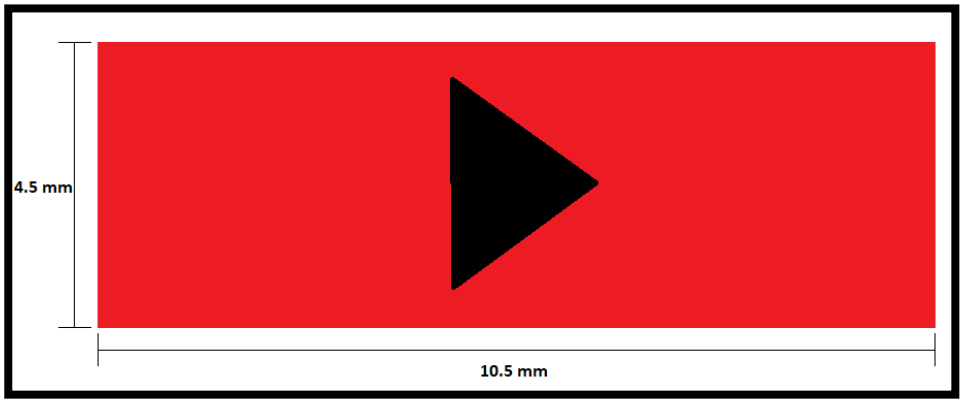
Figura 3 – Principais atividades



Fonte: elaborado pelo autor.

Para utilização da TUI no jogo precisa criar as peças de movimentações e de iniciar a verificação da imagem. Todas devem ter as mesmas dimensões, e recomendasse utilizar 4.5mm x 10.5 mm (Figura 4). A peça que indica para aplicação verificar se existem comandos na imagem e em seguida executar os mesmos, deve estar em uma cor diferente das restantes, como por exemplo, ser na cor vermelha. Também para ajudar as pessoas a identificarem mais facilmente o que faz esta peça é melhor colocar o símbolo do *play*.

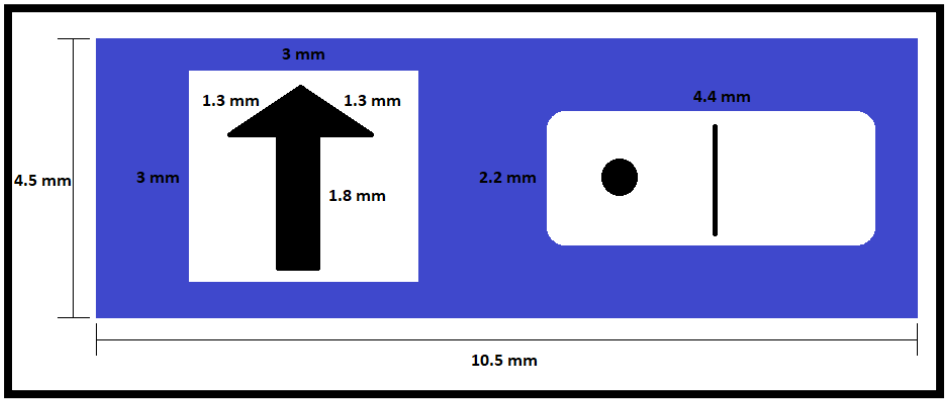
Figura 4 – Peça que indica para aplicação verificar a imagem



Fonte: elaborado pelo autor.

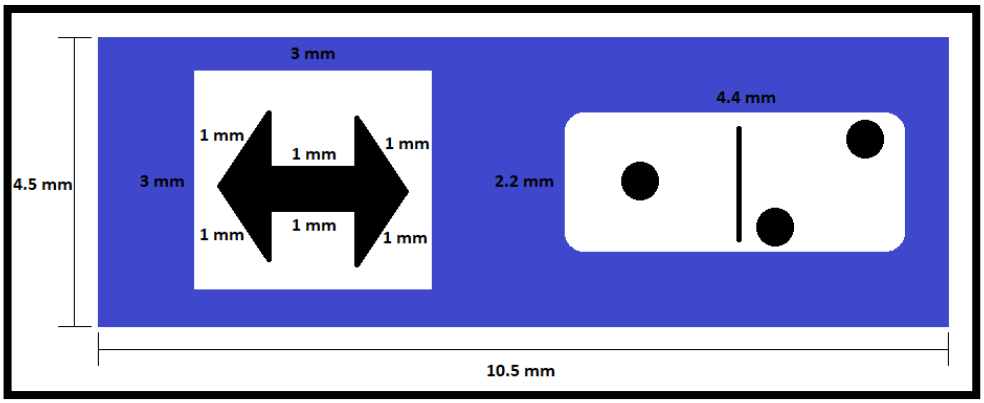
As peças de movimentações (Figura 5) e de repetição (Figura 6) devem ser de cor diferente da falada anteriormente. No canto esquerdo deve ter um papel no formato de um quadrado 3mm x 3mm, contendo no interior uma flecha. Para peça de movimentação a flecha deve ter uma "calda" com 1.8mm para indicação da direção, assim sendo possível alternar as direções apenas rotacionando o quadrado sempre em um ângulo de 90º. Já para a peça de repetição deve ser utilizado uma flecha bidirecional. No lado direito deve ser posicionado o dominó ou um papel no formato de um dominó 2.2mm x 4.4 mm, o mesmo é responsável por indicar a quantidade de vezes que deve ser repetido a direção.

Figura 5 – Peça de movimentação



Fonte: elaborado pelo autor.

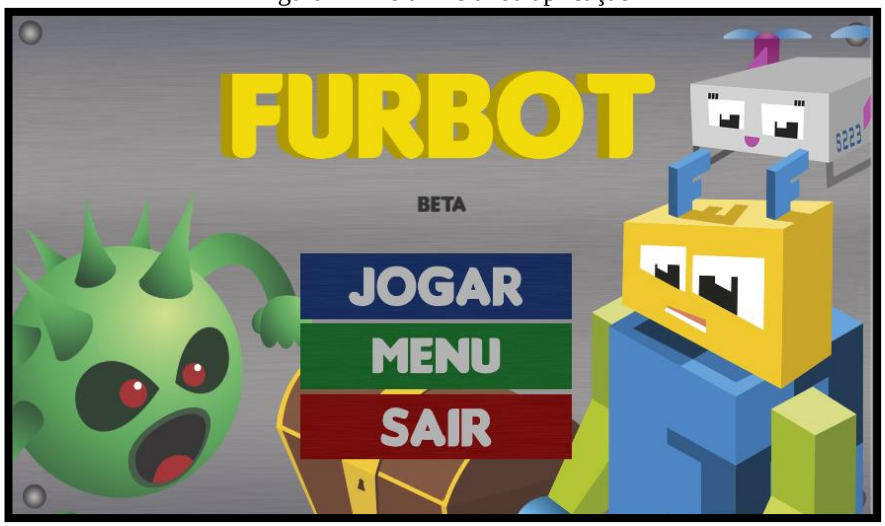
Figura 6 – Peça de repetição



Fonte: elaborado pelo autor.

A tela inicial da aplicação exibe alguns personagens, sendo eles o FURBOT (personagem principal), a S-223 e o alienígena chamado de Buggien (Figura 7). Esta tela fornece três funções, sendo elas: `JOGAR`, `MENU` e `SAIR`. A primeira função, `JOGAR` redireciona para próxima tela, sendo possível ver a tela de tutorial de funcionamento do jogo ou caso esteja utilizando interface tangível vai direto para tela de seleção de fase. A segunda função, `MENU` disponibiliza duas opções de interações, como `Interface Tangível` ou `Interface Gráfica`. A terceira função `SAIR`, encerra a aplicação.

Figura 7 – Tela inicial da aplicação



Fonte: elaborado pelo autor.

Se a função `MENU` for acionada e em seguida a opção de `Interface Tangível`, vai abrir a tela para configuração de identificação da peça principal. Quando aberta, é carregado todos os dispositivos de câmera conectados, também é verificado se já tem alguma configuração realizada e salva anteriormente. Caso seja encontrado é carregada as configurações, sendo assim se a peça principal estiver visível pela câmera é apresentada ao usuário a marcação ao redor da mesma com a cor branco (Figura 8). Existem quatro funções na tela de configuração, sendo elas: configurar o intervalo de cor da peça principal, mudar de câmera, salvar e cancelar alterações. A função de realizar a configuração de intervalo de cor é acionada quando selecionado o local da peça principal no *display* da aplicação. Marcando assim, com branco, a maior área encontrada nas coordenadas selecionadas. Caso a função de mudar de câmera seja pressionada vai ser alternado entre os dispositivos de câmera conectados. Ao acionar a função para salvar vai ser gravada as informações como, o intervalo de cor encontrado anteriormente, o dispositivo que estava utilizando para captura de imagem e a indicação que vai utilizar a TUI para programação do robô. Por fim, a função cancelar volta para tela principal cancelando todas as alterações realizadas anteriormente.

Figura 8 – Tela de configuração da interface tangível

Fonte: elaborado pelo autor.

Após a seleção da fase desejada e se estiver utilizando a interface tangível, são carregadas as configurações realizadas inicialmente carregando a classe `InterfaceTangivel`. No meio do jogo é possível alternar entre os dispositivos acessíveis de captura de imagem e tambem mostra o que está sendo visível pelos mesmos. Caso não esteja em diálogo com o usuário e posicione as peças em frente ao dispositivo de captura de imagem, juntamente com a peça configurada inicialmente, a aplicação verifica se em três frames ocorreu o mesmo conjunto de comandos. Após a identificação de todos os comandos é listando no canto direito a sequência de execução (Figura 9) e iniciado as movimentações.

Figura 9 – Utilização da interface tangível no jogo

Fonte: elaborado pelo autor.

É importante que o usuário posicione o dispositivo de captura da mesma maneira que estão posicionados as peças, pois o ângulo das peças interfere no resultado esperado, como acabar identificando que é para andar para direita, mas o esperado seria estar indo para esquerda. A aplicação faz uma pequena correção no ângulo, tentando assim resolver possíveis problemas futuros.

3.2 IMPLEMENTAÇÃO

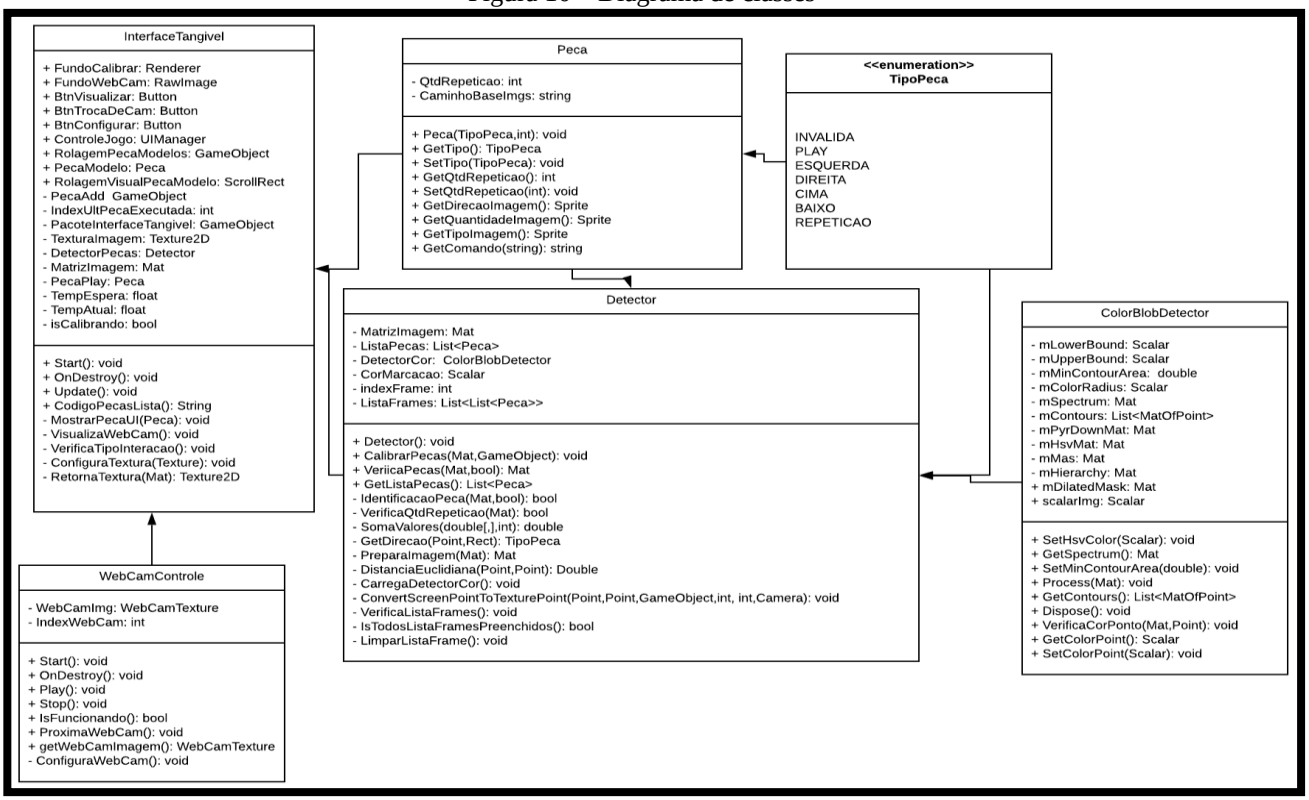
Para desenvolver a integração com interface tangível foi utilizado o Unity, scripts em C#, OpenCV 4.3.0 e Photoshop. Para realizar a integração do Unity com OpenCV foi utilizado o pacote OpenCV For Unity distribuído pela Enox Software (2020). O pacote disponibiliza formas para conversão de matrizes para `Texture2D` (e vice-versa) aceita pelo Unity, e também métodos para processamento e reconhecimento de imagens. O pacote OpenCV For Unity atualmente tem suporte para as seguintes plataformas: iOS, Android, Windows, Linux, MacOS e WebGL. Para acessar os dispositivos e capturar as imagens foi utilizado `WebCamTexture` e `WebCamDivece`. Para criação das imagens utilizadas no projeto como as imagens de representação dos objetos do mundo real, foi utilizado Photoshop distribuído pela Adobe (2020).

Após a detecção das peças é verificado se todas já foram convertidos para linguagem aceita pelo compilador e executado, caso tenha alguma peça que ainda não, é passado por cada uma de baixo para cima convertendo para os comandos aceitos. Também é gerado outro objeto de `Textura2D` para cada um deles. Após isso é adicionado representações digitais na barra de rolagem conforme é executado. Por fim, o compilador roda os comandos encontrados, se ocorrer algum problema como o robô acabar batendo em algo, é mostrado as mensagens de orientações para o usuário.

A Figura 10 exibe o diagrama de classes conforme do projeto. A classe `InterfaceTangivel` é responsável por controlar quando deve realizar a detecção dos objetos, retornar os objetos convertidos para comandos que o compilador aceita, mostrar a representação digital dos objetos do mundo real e por último controlar algumas funções dos componentes em tela. A classe `WebCamControle` realiza a integração com o dispositivo de câmera e retorna a imagem capturada pela mesma. `Peca` é a classe que representa os objetos do mundo real para o mundo digital, sendo assim, nela fica informações como qual tipo de peça, quantidade de vezes que deve realizar determinado comendo e as respectivas imagens do objeto. A classe `ColorBlobDetector` disponibilizada nos exemplos do Opencv, faz a detecção da região de interesse, com base na cor passada como parâmetro, fazendo parte assim da calibração. Por último a classe

`Detector` faz a detecção dos objetos esperados, e é responsável por realizar a calibração e retornar a lista de objetos identificados.

Figura 10 – Diagrama de classes



Fonte: elaborado pelo autor.

Após o usuário entrar na fase desejada e realizar a leitura das orientações dadas pelo jogo, é iniciado o processo de captura das imagens do dispositivo disponível. Inicialmente é identificado a peça base, conforme calibrado anteriormente. O primeiro passo é aplicar o filtro `GaussianBlur` para remoção de ruídos. Em seguida é utilizado o método `pyrDown` duas vezes conforme as duas primeiras linhas do Quadro 4, deixando assim a imagem menor. Próximo passo é realizar a conversão da cor RGB para HSV com a função `cvtColor`, desta forma as cores da imagem ficam mais adequadas para identificação. Em seguida é utilizado a função `iRange` com intervalos de cores inicialmente configuradas, tendo então as regiões de interesse. Após obter as regiões que se encontram no intervalo de cor desejado é utilizada a função morfológica `dilate` para deixar os traços mais destacados, para então ter melhores resultados quando utilizar a função `findContours` na procura dos contornos existentes na imagem.

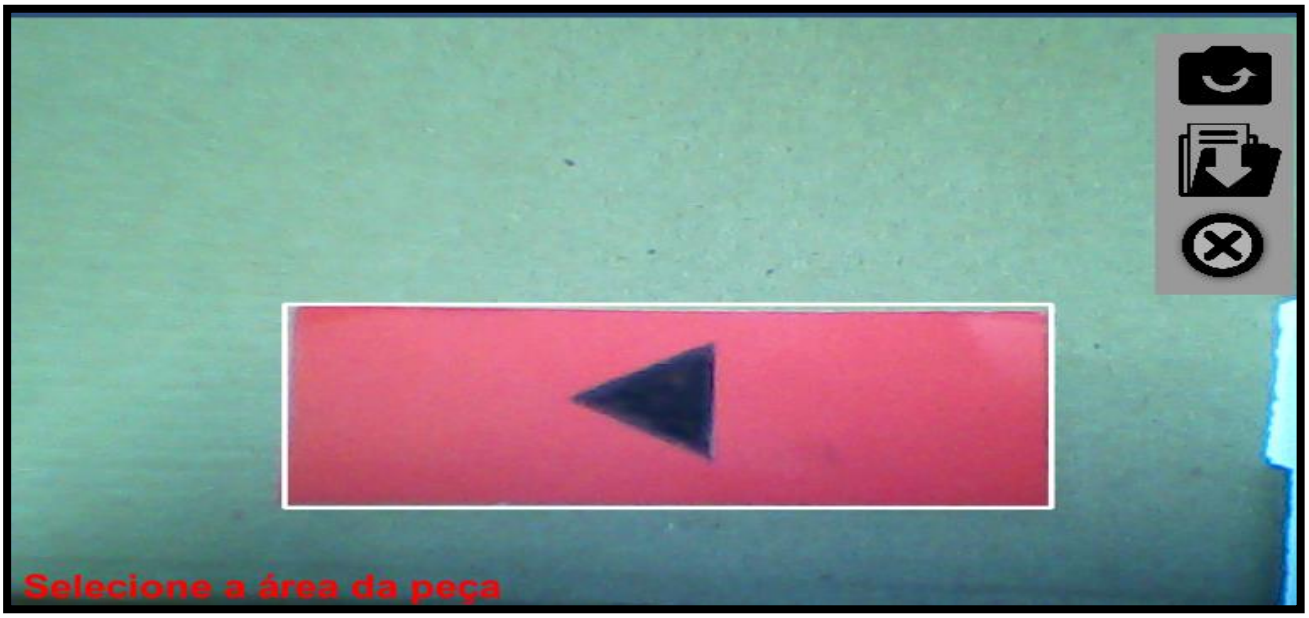
Quadro 4 - Identificação da área de interesse configurada

```
74          Imgproc.pyrDown (rgbaImage, mPyrDownMat);
75          Imgproc.pyrDown (mPyrDownMat, mPyrDownMat);
76          Imgproc.cvtColor (mPyrDownMat, mHsvMat, Imgproc.COLOR_RGB2HSV_FULL);
77          Core.inRange (mHsvMat, mLowerBound, mUpperBound, mMask);
78          Imgproc.dilate (mMask, mDilatedMask, new Mat ());
79
80          List<MatOfPoint> contours = new List<MatOfPoint> ();
81          Imgproc.findContours (mDilatedMask, contours, mHierarchy, Imgproc.RETR_EXTERNAL, Imgproc.CHAIN_APPROX_SIMPLE);
82          double maxArea = 0;
83          foreach (MatOfPoint each in contours) {
84              MatOfPoint wrapper = each;
85              double area = Imgproc.contourArea (wrapper);
86              if (area > maxArea)
87                  maxArea = area;
88          }
89          mContours.Clear ();
90          foreach (MatOfPoint each in contours) {
91              MatOfPoint contour = each;
92              if (Imgproc.contourArea (contour) > mMinContourArea * maxArea) {
93                  Core.multiply (contour, new Scalar (4, 4), contour);
94                  mContours.Add (contour);
95              }
96          }
```

Fonte: elaborado pelo autor.

A seguir a lista de contornos é percorrida buscando a maior área, com o resultado é criado um retângulo. Com isso é possível verificar qual é o ângulo da peça e altera para que fique com ângulo esperado. Após estar alinhado corretamente é criado o retângulo branco ao redor da peça identificada na imagem, para assim o usuário pode identificar o que foi detectado (Figura 11) e salvar caso esteja corretamente a marcação.

Figura 11 – Peça principal configurada



Fonte: elaborado pelo autor.

Com a identificação da peça principal na etapa anterior é utilizado o retângulo para a identificação do restante das peças que podem estar conectadas na parte superior. Percorre-se a imagem subindo o retângulo, alterando então somente o eixo Y, diminuindo com o próprio *height* do retângulo, até que chegue ao limite da imagem ou não retorne nenhuma peça no retângulo que está sendo verificado. A primeira etapa a ser executada depois de obter a parte que se deseja verificar é realizar o pré-processamento como pode ser visto na linha 342 e 343 do Quadro 5. Nesta etapa se aplicou o filtro `GaussianBlur` para suavizar a imagem e retirar os pequenos ruídos, utilizando a máscara 5x5 por apresenta os melhores resultados. O próximo passo foi converter a imagem para escala de cinza e aplicar um filtro de `limiarização` para destacar os contornos da imagem.

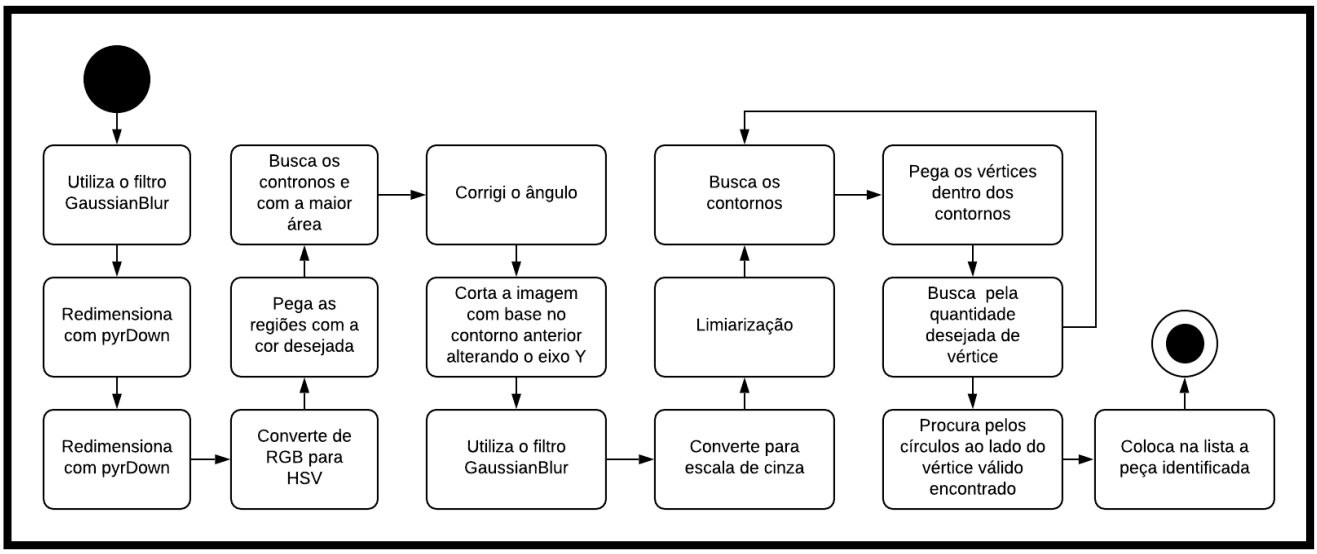
Quadro 5 – Preparação das regiões para verificação

```
342             Imgproc.GaussianBlur(peca, peca, new Size(5, 5), 1);
343             Imgproc.cvtColor(peca, peca, Imgproc.COLOR_RGBA2GRAY);
344             Imgproc.threshold(peca, peca, 0, 255, Imgproc.THRESH_BINARY | Imgproc.THRESH_OTSU);
```

Fonte: elaborado pelo autor.

Após a preparação da região é procurado pelos contornos utilizando a função `findContours`, indicando com o parâmetro `RETR_LIST` que deseja que retorne todos os contornos sem nenhuma hierarquia. Com a listas de contornos obtidos é utilizado a função `approxPolyDP` para obter o número de vértices do contorno. Caso tenha 4 contornos e tenha uma pequena margem de diferença entre o *height* e *width*, será considerado um quadrado e irá chamar a função novamente passando somente a área dentro do quadrado encontrado. Será passado por todo o processo novamente, mas desta vez procurando por 10 vértices, que representa o laço de repetição ou 7 vértices que representa a direção da peça de caminhar. Para identificar qual direção representa a seta é calculado a distância euclidiana entre cada ponto e colocado em um *array* bidimensional, sendo que a linha representa os vértices e a coluna a distância do vértice até os outros. Após ter o *array* preenchido, se calcula para cada linha a soma das distâncias nas colunas, procurando pela linha com o maior número, encontrando os pontos da "calda" da seta que representa a direção. A direção é definida verificando a qual canto do quadrado está mais próximo do ponto da "calda" encontrado, assim definindo a direção oposta. Por fim, se encontrado uma peça válida utiliza a função `HoughCircles`, para identificação de quantas vezes deve ser feito a repetição da peça encontrada, e caso encontre no intervalo de um até doze adiciona a peça a lista. A Figura 12 exibe o fluxo do processamento de imagem descrita acima.

Figura 12 – Fluxo do processamento de imagem para reconhecimento de cada peça



Fonte: elaborado pelo autor.

4 RESULTADOS

Este capítulo está dividido em duas seções para apresentação dos resultados obtidos com o desenvolvimento deste trabalho. A primeira seção apresenta os resultados obtidos inicialmente em relação a programação de reconhecimento de imagem. Já a segunda seção apresenta o comparativo entre o trabalho atual e os trabalhos correlatos.

4.1 RECONHECIMENTO DAS PEÇAS

Se teve uma ideia inicialmente para a identificação das peças, foi utilizado a função `cascade`, pois em outros projetos onde teve há necessidade de detecção de objetos no mundo real, foi obtido bons resultados. Nesta função se utiliza um conjunto de classificadores criados a partir de bases feitas por imagens positivas e negativas. Para criação do classificador vai muito tempo de preparação e treinamento, sendo muito custoso para ser criado um classificador onde tem resultados aceitáveis. Como as peças são criadas pelos usuários, as mesmas podem acabar tendo características diferentes umas das outras, e com isso ter problemas com o classificador criado, invalidando o mesmo.

Em seguida foi alterado para utilizar na identificação a função `inRange` juntamente com a função `Canny`, apresentando então muitos problemas com falsos positivos. Muitas vezes acabava nem encontrando as peças, devido não ter sucesso na procura das bordas da imagem. Para a utilização do detector de borda se utilizou o filtro `gaussianBlur` e as funções morfológicas `erode` e `dilation` para remoção dos possíveis ruídos, mesmo assim, retornou muito ruídos dificultando assim o resultado esperado.

Como as ideias anteriores estavam apresentando muitos problemas foi necessário buscar outra forma que resolvesse o problema da aplicação. Mas como a utilização da função `inRange` mostrou-se ter bons resultados, foi mantida a utilização só que combinada com a função de limiarização `threshold`, onde retorna a imagem com menos ruídos e de uma forma em que ajuda a encontrar os contornos da imagem. Esta última abordagem trouxe resultados melhores que as anteriores, se mostrou ser mais rápida por ter menos processos executados.

4.2 COMPARATIVO ENTRE OS CORRELATOS

O trabalho atual possui os mesmos propósitos que os correlatos, pois todos são para auxiliar no aprendizado escolar. Quando comparado o trabalho atual com o RoboEduc, é possível ver que o trabalho correlato acaba sendo muito mais complexo e cansativo para resolução dos exercícios devido a programação ser textual ou por interação no *display* do dispositivo, quanto ao trabalho atual que utiliza Interface de Usuário Tangível. O Coding Awbie tem o funcionamento semelhante ao trabalho atual, pois através das peças posicionadas em frente ao dispositivo de captura de imagem, é movimentado o personagem. Só que para utilização do Coding Awbie é necessário a compra de um kit de peças, já o trabalho atual pode ter as peças criadas pelos próprios alunos com o auxílio do professor. Já o software VisEdu-CG é semelhante ao trabalho desenvolvido por ter uma programação das ações predeterminadas, onde as peças têm um posicionamento específico para serem posicionados, para assim serem consideradas válidas.

5 CONCLUSÕES

O trabalho implementou todos os objetivos específicos, e por ser um aplicativo para plataforma Android, acaba facilitando a possibilidade de utilização nos dias de hoje. A aplicação é capaz de realizar o reconhecimento das peças criadas pelo o usuário nas imagens capturadas, passando assim a executar as ações solicitadas pelos usuários e

mostrando a representação digital de cada comando. Devido a algumas limitações como a luz ambiente, sombras feitas pelo usuário, o local em que as peças ficam posicionadas, ângulos e a forma que são feitas as peças, os comandos executados podem acabar nem sempre sendo os desejados. Sendo assim é importante que sejam tomados cuidados em relação ao ambiente que será rodada a aplicação e que sejam seguidos corretamente as recomendações apresentadas no Apêndice A, para criação das peças de representação dos comandos aceitos, prevenindo assim a execução de comandos indesejados.

Devido a utilização de interface tangível a jogabilidade do jogo se tornar mais atrativa e de fácil entendimento ao usuário que a utilizada por padrão no Furbot, podendo assim ser utilizada com crianças do ensino fundamental. Mesmo a aplicação tendo algumas limitações, é possível ser utilizada nos ambientes escolares onde as limitações estejam sobre controle. Como sugestões de possíveis opções de extensão para este trabalho são:

a) melhorar a forma de interação de quando devem ser executados os comandos;
b) adicionar todos os comandos que são possíveis para execução quando não se utiliza interface tangível;
c) minimizar as limitações de ambiente;
d) diminuir o tempo para o reconhecimento das peças;
e) melhorar para os ângulos não interferirem na qualidade do reconhecimento;
f) permitir o usuário cadastrar peças novas.

REFERÊNCIAS

CASTRO, Viviane Gurgel. **RoboEduc**: especificação de um software educacional para ensino da robótica às crianças como uma ferramenta de inclusão digital. 2008, 182 f. Dissertação (Mestrado em Engenharia da Computação) - Universidade Federal do Rio Grande do Norte. Natal.

CONFORTO, Debora et al. **Pensamento computacional na educação básica**: interface tecnológica na construção de competências do século XXI. Revista Brasileira de Ensino de Ciências e Matemática, v. 1, n. 1, 10 ago. 2018.

DINIZ, Sirley Nogueira de Faria. **O uso das novas tecnologias em sala de aula**. 2001. 172 f. Dissertação (Mestrado em Engenharia de Produção) - Universidade Federal de Santa Catarina, Florianópolis.

IGNÁCIO, Wagner. **O pensamento computacional na educação brasileira e o papel das instituições de ensino tecnológico**. 2018. 43 f. Trabalho de Conclusão de Curso (Especialização) - Universidade Federal de Juiz de Fora, Juiz de Fora.

JORGE, Fábio Rodrigues. **Interação de realidade aumentada mobile com interfaces tangíveis tabletop**. 2012. 47 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro Universitário Eurípides de Marília. Marília.

KOCH, Marlene Zimmermann. **As tecnologias no cotidiano escolar**: uma ferramenta facilitadora no processo ensino-aprendizagem. 2013. 36 f. Trabalho de Conclusão de Curso (Especialização em Gestão Educacional) - Universidade Federal de Santa Maria, Sarandi.

MATTOS, Mauro et al. **Furbot Móvel**: um jogo para o ensino do pensamento computacional. Anais dos Workshops do Congresso Brasileiro de Informática na Educação, [S.l.], p. 1294, nov. 2019. Disponível em: <https://www.br-ie.org/pub/index.php/wcbie/article/view/9091>. Acesso em: 21 jul. 2020.

MONTIBELER, James Perkison. **VisEdu-CG:** aplicação didática para visualizar material educacional, módulo de computação gráfica. 2014. 105 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

OSMO. **Coding Awbie**. [S.I.], [2015]. Disponível em: <https://www.playosmo.com/en/coding/>. Acessado em: 5 jul. 2020.

OTTO, Patrícia Aparecida. **A importância do uso das tecnologias nas salas de aula nas series iniciais do ensino fundamental I**. 2016. 18 f. Trabalho de Conclusão de Curso (Pós-graduação em Educação na Cultura Digital) – Universidade Federal de Santa Catarina, Florianópolis.

SANT' ANNA, Andrew de Castro; FERRONATO, Ana Carolina Clivatti. **Interfaces Naturais e Interfaces Tangíveis**. [2017]. Disponível em: http://www.professores.uff.br/screspo/wp-content/uploads/sites/127/2017/09/artigoIHC2.pdf. Acesso em: 29 jun. 2020

SILVA, Josiel Moreira. **Estimulando o Pensamento Computacional com Jogos Digitais:** uma abordagem utilizando *Scratch*. 2018. 128 f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Software) – Universidade Federal do Rio Grande do Norte. Natal.

THOALDO, Deise Luci P.B **O uso da tecnologia em sala de aula**. 2010. 35 f. Monografia (pós-graduação em gestão pedagógica; educação infantil e series iniciais) – Faculdade de Ciências Humanas, Letras e Artes da Universidade Tuiti do Paraná, Curitiba.

TOLENTINO, Paula Cristina de Souza. **Influência das novas tecnologias na educação fundamental**. 2013. 32 f. Trabalho de Conclusão de Curso (Especialização) – Universidade Tecnológica Federal do Paraná, Medianeira.

WING, Jeannette Marie. **Pensamento computacional**. [2006]. Disponível em: https://www.cs.cmu.edu/afs/cs/usr/wing/www/ct-portuguese.pdf. Acessado em: 21 jun. 2020.