PETRI DOG

Aplicação Lúdica para Desenvolvimento da Lógica
e Programação com Crianças

Mestrado em Design e Multimédia
Faculdade de Ciências e Tecnologias
Universidade de Coimbra

Joana Margarida Teixeira Ribeiro

Julho de 2017

Mestrado em Design e Multimédia
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

Orientação
Licínio Gomes Roque
Maria José Marcelino

Júri
Tiago José dos Santos Martins da Cruz
Bruna Raquel Santos Sousa

Resumo

Actualmente, os dispositivos móveis estão presentes na maioria das actividades humanas. O tablet tornou-se um dispositivo comum, sendo utilizado para tirar fotografias, jogar, criar documentos, aceder à Internet, entre outros. Devido ao acesso à Internet, muitas são as aplicações descarregadas e utilizadas todos os dias, sendo desenvolvidas para os mais diversos fins.

Uma das áreas de desenvolvimento de aplicações é a área da educação. Tendo contacto com estes dispositivos desde cedo, as crianças sentem-se à vontade para experimentar e explorar tudo o que as aplicações têm para lhes dar. Muitas aplicações são desenvolvidas com o intuito de ensinar e promover o desenvolvimento de capacidades, promovendo a transmissão de conhecimento de uma forma fácil e descontraída. Dentro da área da educação, têm sido desenvolvidas diversas ferramentas de aprendizagem direccionadas para o ensino da programação em crianças.

Esta dissertação foca o processo de design e avaliação de uma aplicação para tablet que tem como função proporcionar o desenvolvimento do raciocínio associado à programação e resolução de problemas. Através de um jogo com um enredo simples e com vários níveis, pretende-se dar a estimular uma forma de pensar subjacente à programação, ao mesmo tempo que este desenvolve capacidades, como a resolução de problemas, importantes para o dia-a-dia.

Palavras - Chave

Programação, Resolução de Problemas

Abstract

Nowadays, mobile devices are used in our everyday tasks. The tablet has become a day-to-day device used to take photographs, play games, write documents, access the Internet, among many other things. Due to the easy Internet access, thousands of applications are downloaded everyday to fulfill many different needs.

One of the areas in application development is Education. Being in contact with this technology from a very young age, children tend to feel comfortable with experimentation and exploration of every kind of application. Many applications are built to teach and develop skills, promoting knowledge in an easy and relaxed way. In the eld of Education many applications have been developed to teach children how to write code.

This thesis is about the creation of an application to tablet devices which aims to help the development of logic applied to programming and solving problems. Through the game development, using a simple plot and a level progression system, it is intended to give the player knowledge of programming rules and elements, and at the same time helping to improve their problem solving skills.

Keywords

Programming, Problem Solving

Agradecimentos

Agradeço ao meu orientador, Licínio Roque, por ter aceitado a minha proposta de dissertação, acreditando desde início no seu potencial, e pela ajuda e apoio prestados durante a concretização da mesma.

Agradeço à minha família, em especial aos meus pais e irmã, pelo apoio e encorajamento ao longo de todo o percurso e pela paciência nos momentos mais difíceis.

Agradeço ao Iuri e ao Pedro todas as críticas e sugestões, assim como a amizade e apoio que me deram durante o desenvolvimento do projecto.

Agradeço a todas as pessoas que se disponibilizaram a experimentar a aplicação, em especial à Sofia, à Catarina e ao Zé pelo apoio e amizade, e à Ana Garcês por me ter ajudado a encontrar crianças dispostas a jogar o Petri Dog.

Obrigada.

Índice

Lista de Figuras

1 - Introdução

Os níveis de insucesso obtidos em disciplinas como a Matemática e a Programação têm motivado a pesquisa e desenvolvimento de ferramentas que estimulem o gosto pelo raciocínio lógico. A falta de desenvolvimento desta capacidade é o que leva, muitas vezes, os alunos a reprovarem a Matemática. Quer seja por não entenderem o problema, quer seja por não conseguirem arranjar uma solução para o resolver.

Muitas vezes, o método de ensino aplicado nas escolas não será o mais indicado para que esta componente lógica possa ser desenvolvida. Actualmente, grande parte do ensino está centrado na aprendizagem através da utilização da memória, o que faz com que os alunos decorem a matéria, coloquem essa informação nos testes, e pouco tempo depois esqueçam o que aprenderam. Este método de ensino não está orientado para desenvolver uma competência importante, que é necessária para a vida futura de qualquer jovem - a resolução de problemas.

Relativamente à aprendizagem da programação, os alunos estão acostumados ao método de ensino aplicado nas escolas e não são preparados para estudar através da resolução de exercícios. O não desenvolvimento da capacidade lógica e da resolução de problemas leva a que tenham dificuldade em interpretar e resolver novos problemas colocados, assim como o ensino da programação ser focado na sintaxe também não permite uma fácil compreensão do propósito da programação.

De forma a estimular o gosto pelo pensamento lógico em crianças e jovens, foi proposta a concepção de uma aplicação para tablet que permite o desenvolvimento desta componente. Partindo de exemplos de outras ferramentas de aprendizagem já desenvolvidas, foi criado o jogo Petri Dog, onde o jogador é levado a construir regras para o personagem de forma a este conseguir ultrapassar os obstáculos que lhe são colocados no caminho.

Esta ferramenta permitirá aos jovens adquirir algumas competências que não são facilmente desenvolvidas no contexto escolar. Com o fomento da prática de raciocínio lógico pretende-se ajudá-los no seu percurso académico (ou fora dele) e melhorar, assim, os seus níveis de sucesso também noutras disciplinas, como a matemática. No caso da programação, esta ferramenta permitirá entender mais facilmente a lógica subjacente à actividade de programação e facilitará a interpretação e resolução de problemas.

1.1. Motivação

A Matemática, tal como a programação, pretende que os alunos aprendam a pensar e a resolver problemas. Estas duas áreas de estudo requerem um estudo baseado na prática intensiva, na compreensão do problema e na reflexão sobre o mesmo.

Uma vez que existe insuficiente desenvolvimento da lógica e da resolução de problemas no meio escolar, é importante arranjar outros meios que a consigam combater. Sendo os tablets, muitas vezes, utilizados para entreter e ensinar as crianças através da utilização de jogos e aplicações, foi idealizada uma forma de juntar a diversão de um jogo ao ensino e desenvolvimento de capacidades importantes para a criança.

Desenvolver uma aplicação para dispositivos móveis, que contribui para o crescimento e desenvolvimento das crianças, é um enorme factor de motivação. Para além da possibilidade de desenvolver conhecimentos na área do desenvolvimento de aplicações para dispositivos móveis, também é proporcionado um desenvolvimento do conhecimento na área de design.

1.2. Público-Alvo

O público-alvo desta aplicação são as crianças a partir dos 8 anos, idade em que começam a desenvolver a capacidade de utilizar o pensamento lógico para resolver problemas concretos. No entanto, esta aplicação pode ser utilizada por indivíduos de qualquer idade que desejem desenvolver a capacidade da lógica e de resolução de problemas.

1.3. Objectivos

Uma vez que a aprendizagem da programação é dificultada devido à falta de capacidades na resolução de problemas, demonstrada por uma parte significativa de alunos que iniciam o estudo da programação, o objectivo desta dissertação é desenvolver uma aplicação lúdica que proporcione o desenvolvimento da lógica e programação em crianças a partir dos 8 anos.

Esta ferramenta de aprendizagem tem como objectivo levar o jogador a resolver problemas, desafiando-o a construir uma solução para eles. É através da construção de regras para o personagem que o jogador consegue levá-lo a ultrapassar os obstáculos que lhe são colocados no caminho.

Nesta ferramenta de aprendizagem desenvolvida, o foco está na criação de motivação para que o jogador sinta vontade de continuar a aprender; preparar o jogador para uma aprendizagem mais aprofundada da programação no futuro; conseguir que o jogador desenvolva a área de resolução de problemas e de lógica como parte de um exercício lúdico estimulante; e desafiar o jogador ao longo do jogo através da introdução de desafios cada vez mais complexos.

1.4. Metodologias

O desenvolvimento deste projecto teve por base a metodologia Design Science Reasearch (figura 1), um método de pesquisa que tem como objetivo desenvolver conhecimento necessário à concepção de artefactos eficazes.

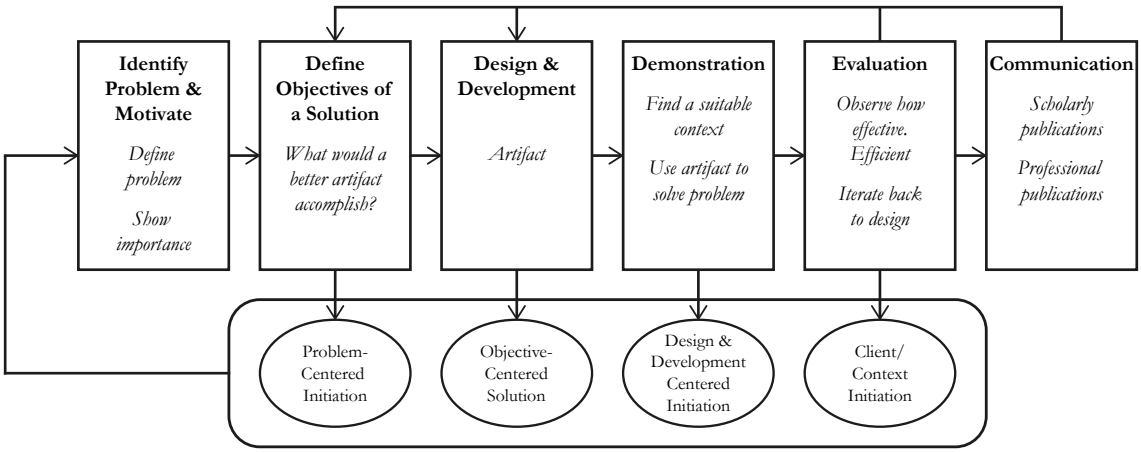


Figura 1. Design Science Research Methodology [61]

Inicialmente, foi identificado o problema e realizadas pesquisas sobre jogos e aplicações existentes no mercado, desenvolvidos com o propósito de resolver problemas iguais ou semelhantes ao colocado neste projecto. Desta pesquisa foi possível tomar conhecimento do que já existe, quais as estratégias utilizadas, e quais os aspectos positivos e negativos existentes em cada ferramenta.

Com o conhecimento do que já foi desenvolvido, assim como de algumas matérias importantes relacionadas com a aprendizagem, foi possível definir objectivos e criar uma ideia de solução para o problema colocado.

Para a elaboração do conceito do jogo foi, inicialmente, realizado um estudo sobre a construção de Petri Nets, de modo a conseguir incorporar este método de modelação de sistemas na construção da solução do jogo. Após encontrar um modo de incorporar as Petri Nets, foi

definido o conceito e, posteriormente, as mecânicas do jogo, as regras, o cenário, os desafios, os personagens e os *feedbacks* existentes no jogo.

Com as mecânicas do jogo definidas, foi desenvolvido um protótipo em papel e realizados *design walkthroughs* com base em ensaios de utilização simulada. Após estes ensaios, procedeu-se ao desenvolvimento gráfico da aplicação e, de seguida, foi iniciada a implementação de um protótipo funcional.

Após a implementação do protótipo, foram realizados testes de usabilidade e, após análise dos resultados obtidos, foram efectuadas as alterações necessárias.

1.5. Plano de Trabalho

Inicialmente, foi definido um plano de trabalho para a realização da dissertação (figura 2), que sofreu alterações no decorrer do projecto (figura 3). Estas alterações deveram-se à necessidade de realizar alterações em tarefas dadas como concluídas na primeira entrega, à introdução de novas tarefas e à alteração do tempo de execução de tarefas.

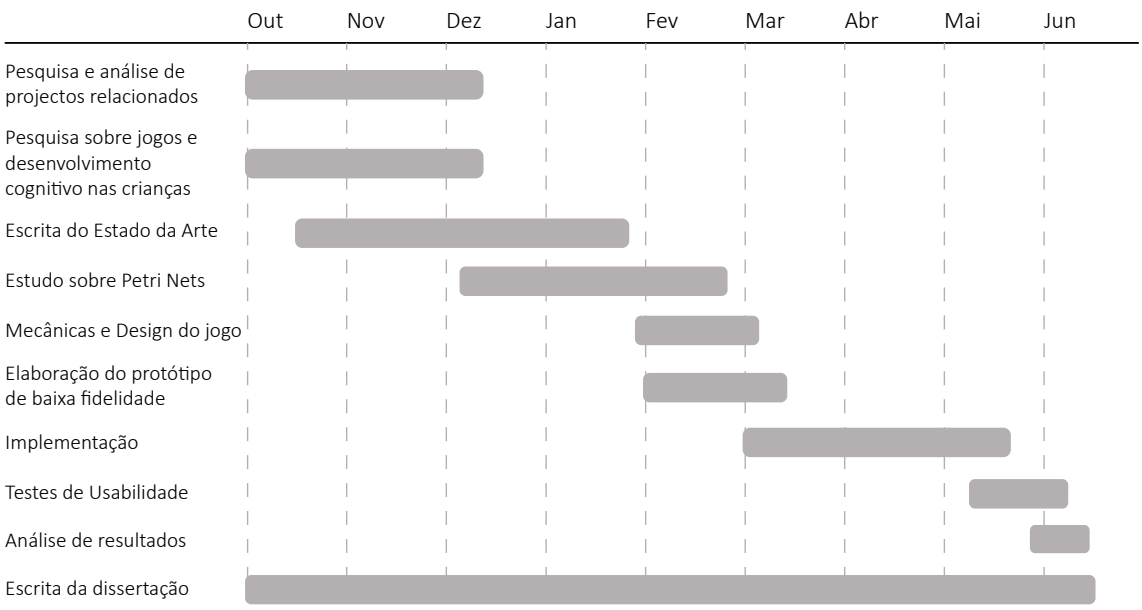


Figura 2. Primeiro plano de trabalho

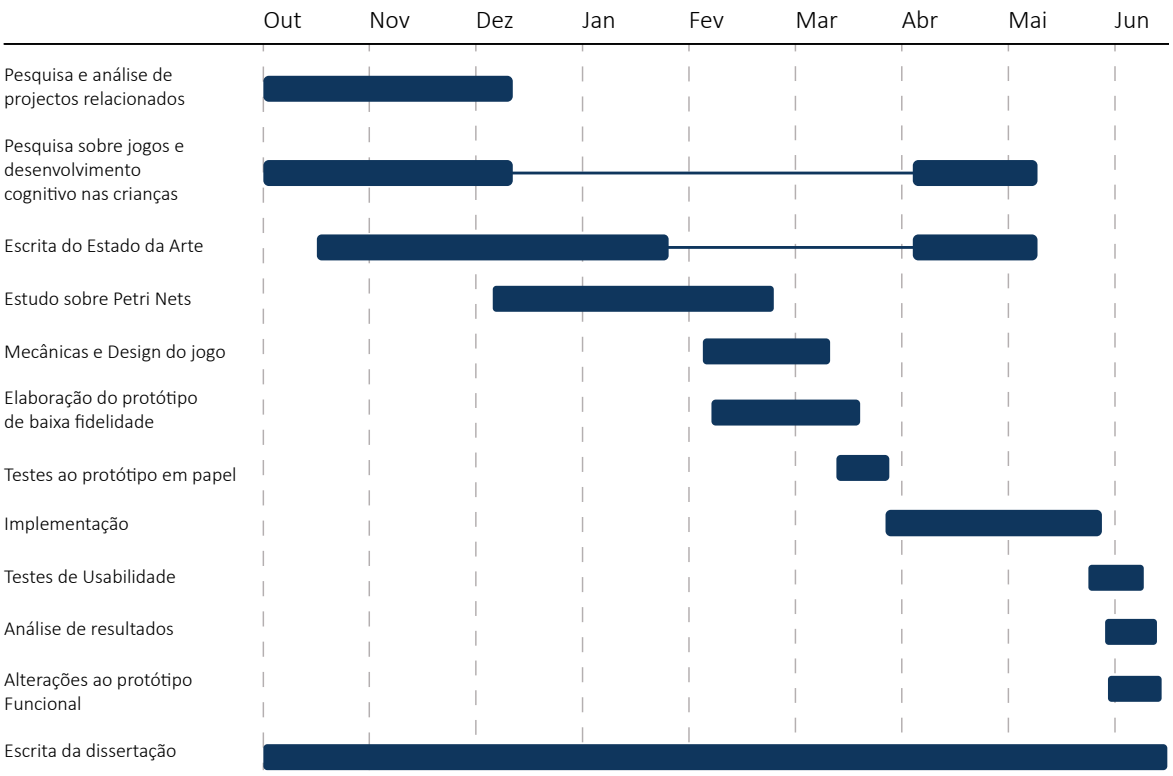


Figura 3. Plano de trabalho final

1.6. Estrutura do Documento

Este documento está dividido em capítulos, organizados da seguinte forma:

O primeiro capítulo apresenta uma *Introdução* do documento, sendo feita uma breve explicação do tema do projecto, as motivações que levaram ao seu desenvolvimento, os objectivos, as metodologias utilizadas na sua realização, o plano de trabalho e a estrutura do documento.

O segundo capítulo é dedicado ao *Estado da Arte*, apresentando a pesquisa realizada sobre ferramentas de aprendizagem já desenvolvidas e outras matérias importantes para o desenvolvimento do projecto.

O terceiro capítulo corresponde à *Proposta de Design*, onde é realizada uma descrição do jogo, definido o conceito e as mecânicas do mesmo, assim como o método utilizado para a construção da solução no jogo.

O quarto capítulo é dedicado ao *Design da Aplicação*, apresentando o protótipo de baixa fidelidade realizado em papel, seguido da estrutura da aplicação e dos elementos e cores que compõem a mesma.

O quinto capítulo corresponde à *Implementação* do protótipo funcional, compreendendo a estrutura da aplicação e uma explicação sobre as funcionalidades implementadas no jogo.

O sexto capítulo é dedicado à *Avaliação de Usabilidade*, explicando de que modo consistiram os testes, quais os resultados obtidos e que alterações foram realizadas ao protótipo funcional após análise dos resultados.

O sétimo capítulo apresenta uma *Conclusão* do documento, onde é realizada uma análise da realização da dissertação e apresentado o trabalho futuro.

Pode fazer download da aplicação Petri Dog através do seguinte link:
https://docs.google.com/uc?export=download&id=0B7nLgd-0EI13NG1XOU5CZXVXeDQ

2 - Estado da Arte

Neste capítulo são abordados temas importantes para o desenvolvimento deste projecto, estando dividido em cinco sub-capítulos:

O primeiro sub-capítulo é dedicado às ferramentas de aprendizagem da programação já desenvolvidas até ao momento. Neste sub-capítulo são apresentadas algumas aplicações existentes no mercado, realizando uma análise sobre o modo de construção da solução que utilizam, quais os elementos da programação que introduzem ao jogador e quais as características que as diferenciam.

O segundo sub-capítulo aborda o desenvolvimento cognitivo da criança, com o objectivo de perceber qual a idade em que a criança começa de desenvolver a lógica, sendo referenciadas as teorias de Jean Piaget e Jerome Bruner sobre o tema.

O terceiro capítulo é referente à aprendizagem baseada em problemas, explicando em que consiste este tipo de aprendizagem e de que modo se difere do método de aprendizagem tradicional.

O quarto capítulo é dedicado ao *Play* e *Game*, explicando de que forma é que estes dois termos se interligam e quais as características de um jogo que fazem com que o jogador se sinta motivado a continuar a jogar.

O quinto capítulo é referente aos *Serious Games*. Neste sub-capítulo são apresentando diversos exemplos de jogos deste género e apresentadas algumas diferenças entre este tipo de jogos e os jogos de entretenimento.

2.1. Ferramentas de Aprendizagem da Programação

Seymour Papert, em 1960, acreditava que as crianças podiam aprender, desenvolver a critividade, inovar e construir um pensamento computacional através da utilização de computadores como ferramentas de aprendizagem [60].

Com base nesta sua crença, nasce a primeira linguagem de programação com o objectivo de se tornar numa ferramenta de aprendizagem - LOGO. Após o aparecimento desta linguagem, outros ambientes foram desenvolvidos com o intuito de promover a aprendizagem da programação em crianças.

2.1.1. LOGO

Logo é uma linguagem de programação visual, criada em 1967. Foi desenvolvida com o objectivo de se tornar numa ferramenta para a aprendizagem da programação, tendo como características principais a modularidade, extensibilidade, interatividade e flexibilidade [1].

Logo é uma linguagem versátil, podendo ser utilizada por jovens que pretendam iniciar-se na programação, ou por alguém experiente na área que pretenda construir projectos mais complexos. O ambiente Logo é caracterizado pela presença de um objecto, representado pela figura de uma tartaruga, que é deslocado no espaço através da introdução de comandos simples. Os comandos, inseridos pelo utilizador, indicam a direcção e a distância que a tartaruga deve percorrer (figura 5).

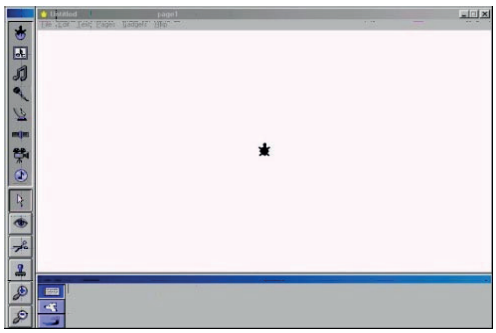


Figura 4. Ambiente *Logo* [39]

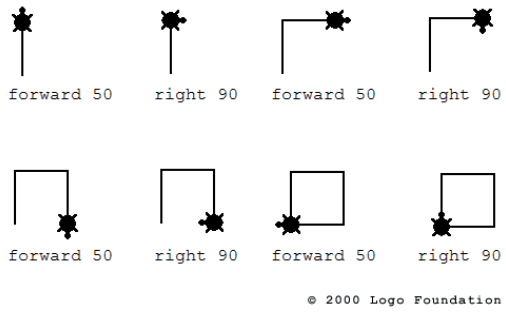


Figura 5. Comandos e respectivo resultado [40]

No processo de desenhar uma determinada figura, o utilizador acaba por desenvolver a lógica e a resolução de problemas, refletindo e modificando o seu código caso o resultado desenhado pela tartaruga não esteja a corresponder ao esperado. Utilizando o Logo, o utilizador explora vários conceitos, tais como: noções espaciais, sequência, estimativa, reversibilidade, operações aritméticas (adição e subtracção) e geometria [2].

A linguagem Logo é o precursor de todos os ambientes de aprendizagem da programação desenvolvidos até aos dias de hoje.

Semelhante ao Logo, foi desenvolvida uma aplicação para iOS que utiliza uma tartaruga como personagem principal, que se move no espaço através de indicações que o utilizador vai introduzindo no ecrã. Com o design mais sofisticado e apelativo ao público mais jovem, este software tem como objectivo introduzir crianças, a partir dos 5 anos, na programação dando-lhes os conceitos base através de desafios que as mesmas têm de superar.

Ao contrário do Logo, Move the Turtle [3], utiliza símbolos em substituição do código escrito (figura 7). Desta forma, torna-se mais simples para as crianças manipular a tartaruga. Há também a introdução de desafios, em que a criança deve levar a tartaruga até ao diamante que aparece numa dada posição do ecrã. Os desafios também trazem algo de positivo, uma vez que leva a criança a ter de resolver problemas e utilizar o raciocínio lógico para alcançar o objectivo estabelecido.

Na aplicação também existe um modo livre, que permite explorar todas as capacidades desenvolvidas ao longo dos desafios, e que dá a possibilidade de criar formas geométricas tal como na ferramenta Logo.

Figura 6. Ambiente *Move The Turtle* [41]

Figura 7. Construção da solução do problema (lado esquerdo) [41]

2.1.2. Scratch

Criado em 2007, Scratch (figura 8) é um ambiente de programação visual, que tem como objectivo introduzir a programação naqueles que não têm qualquer conhecimento na área. Tendo como público-alvo crianças entre os 8 e os 16 anos, o Scratch permite criar jogos, animações e histórias interactivas [4]. De modo a incentivar a auto-aprendizagem, o ambiente de programação do Scratch permite ao utilizador receber um *feedback* imediato, à medida que vai construindo e executando o seu código.

O ambiente do Scratch é dividido em quatro janelas principais: a janela de visualização, utilizada para colocar e editar as *sprites*, assim como para visualizar o resultado da execução do código construído; a janela com as *sprites*, onde são apresentadas as miniaturas de todas as *sprites* utilizadas no projeto, estando em destaque a *sprite* seleccionada num dado instante; a janela de comandos, que possui todos os comandos que o utilizador pode utilizar na construção do seu código; e a janela de construção, para onde o utilizador arrasta os blocos e constrói o seu código. Os comandos disponíveis para a construção do código estão divididos em 8 categorias, permitindo um fácil acesso e utilização dos mesmos por parte do utilizador.

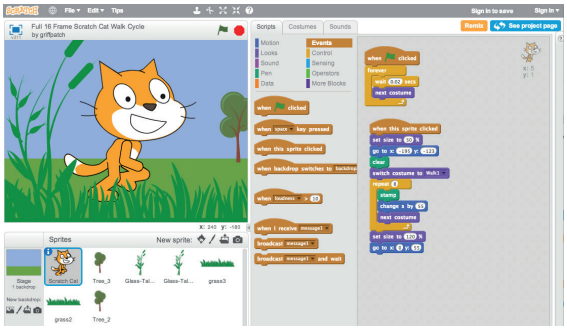


Figura 8. Ambiente *Scratch* [42]

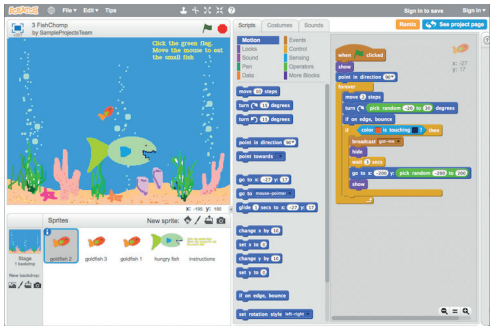


Figura 9. Construção do código (lado direito) [43]

Seguindo a mesma lógica de construção de código do Scratch, foram surgindo, posteriormente, outros ambientes de programação que utilizam blocos com instruções para facilitar a construção do código.

2.1.3. Minecraft: Hour of Code

Minecraft: Hour of Code é um jogo online que, tal como o Scratch, permite utilizar blocos com instruções para construir código (figura 10). Desenvolvido para ser jogado por crianças a partir dos 6 anos, este jogo estimula a aprendizagem da programação, encorajando o jogador a navegar, criar e explorar o mundo 2D do Minecraft e a completar missões através da construção de código. O Minecraft é um jogo mundialmente conhecido e bastante apreciado pelos mais jovens, o que faz com que haja uma motivação extra para se experimentar este jogo [6].

Ao todo existem 14 níveis, ao longo dos quais vão sendo abordadas diversas componentes da programação e a complexidade dos desafios vai aumentando à medida que o jogador passa de nível. O número de blocos disponibilizados em cada nível é o necessário para que o jogador consiga alcançar o objectivo nesse nível. Deste modo, o jogo torna a aprendizagem mais fácil, permitindo que esta se torne gradual.

O Minecraft: Hour of Code permite ao utilizador obter *feedback* aquando da execução do código construído (figura 11). O personagem escolhido pelo jogador executa as acções contidas no código, sendo delineado, ao mesmo tempo, o bloco de código correspondente à acção que está a ser apresentada. Deste modo o jogador associa uma acção a um bloco de código e é capaz de corrigir facilmente possíveis erros.
No final, existe um nível livre que permite ao jogador aplicar os conhecimentos adquiridos ao longo do jogo, tendo disponível todos os blocos de instruções e podendo explorar à sua vontade.

O Minecraft: Hour of Code permite ao jogador aceder ao código JavaScript que está por detrás do código que ele construiu utilizando os blocos com instruções.

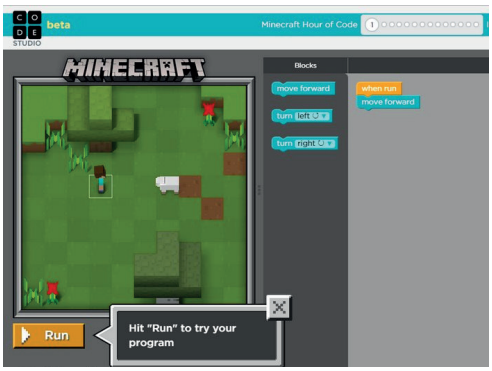


Figura 10. Ambiente *Minecraft: Hour of Code* [44]

Figura 11. *Feedback* representado pelo delineamento a amarelo na área do código que está a ser executado [45]

2.1.4. Daisy The Dinosaur

Daisy The Dinosaur é uma aplicação para iPad que pretende introduzir as crianças na programação através de pequenos desafios (figura 12). Daisy é um dinossauro que executa diferentes acções, sendo estas executadas de acordo com o código construído pelo jogador, utilizando blocos com instruções [9].

Esta aplicação dá ao jogador a opção de jogar em modo de jogo livre ou em modo de desafio. No modo de jogo livre, o jogador pode escolher entre nove comandos diferentes, utilizando-os para criar uma animação com a Daisy. No modo de desafio, o jogador tem de construir um código de forma a resolver o problema que lhe é colocado, utilizando os blocos que lhe são disponibilizados (figura 13). Neste modo, o desafio para o jogador é descobrir como organizar os comandos disponíveis para completar a tarefa atribuída.

A dificuldade dos desafios vai aumentando à medida que o jogador vai avançando no jogo, de modo a tornar a aprendizagem gradual. Em ambos os modos, é apresentada uma seta que dá a indicação visual do local para onde o jogador deve arrastar os blocos, ajudando, assim, o jogador a perceber o funcionamento do jogo. Algumas acções podem ter mais do que uma opção, tal como a do movimento, que pode ser especificado como para a frente ou para trás, e aparecem como um menu *drop-down* no próprio bloco.

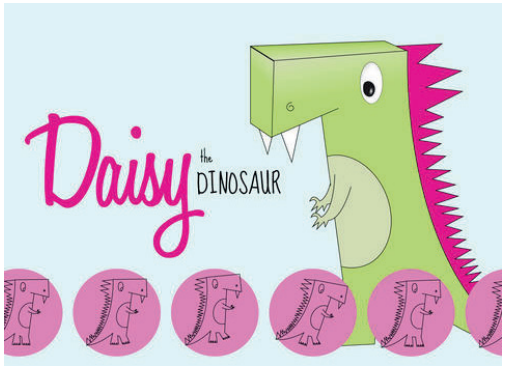


Figura 12. *Daisy The Dinosaur* [46]

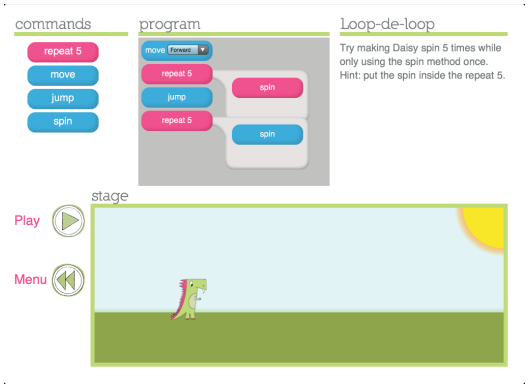


Figura 13. Ambiente *Daisy The Dinosaur* [47]

Outros ambientes de aprendizagem da programação foram surgindo no mercado, diferenciando-se pela utilização de um método de construção de código distinto do utilizado pelas aplicações anteriormente referidas.

2.1.5. Kodu

Lançado em 2009, e disponível para Xbox e PC, Kodu é uma linguagem de programação visual, que foi desenvolvida num ambiente tridimensional, e o seu principal objectivo é estimular a aprendizagem lúdica da programação orientada a objectos. Inspirado na robótica, este software permite criar personagens e objectos, programando-os individualmente, de modo a interagirem com o mundo em redor [10].

A produção de código é feita através de ícones, onde as instruções são descritas e divididas em condições e acções (*When-Do*), e é interligando ícones de código que se formam as regras que definem os comportamentos de um personagem/objecto. Os programas são expressos em termos físicos, utilizando conceitos como visão, audição, e tempo para controlar o comportamento do personagem.

Para a criação do seu próprio jogo, o utilizador começa por criar o seu mundo e, posteriormente, coloca alguma terra e decora-a com objectos inanimados, tais como: edifícios, árvores, pedras, lagos, etc. De seguida, escolhe os personagens que deseja ter no seu jogo, e coloca-os no mundo anteriormente criado.

Após escolher os personagens, pode-se começar a construir instruções para os mesmos, de modo a interagirem com o mundo de uma determinada maneira (figura 15). Selecionando o personagem/objecto desejado, o utilizador cria uma regra seleccionando a situação e a acção que será executada nesse caso. Por exemplo, quando o personagem for contra uma árvore deverá fazer uma determinada acção. De seguida, selecionamos a acção que deve ser executada. Por exemplo, o personagem muda de cor.

Neste ambiente, a programação é feita utilizando ícones e as crianças nem se apercebem que estão a programar pois não estão a utilizar texto nem blocos com instruções que se interligam entre si. Utilizam apenas botões para dizer o que querem que o objecto faça numa determinada situação (figura 14).

Figura 14. Ambiente *Kodu* [48]

Figura 15. Construção de regras para um personagem/objecto [49]

2.1.6. Cargo-Bot

Cargo-Bot é uma aplicação desenvolvida para introduzir os jovens na programação e está disponível apenas para iPad. Nesta aplicação existem seis níveis: Tutorial, Fácil, Médio, Difícil, Crazy e Impossível, sendo cada um deles composto por seis puzzles diferentes [14]. O jogador conquista entre 1 a 3 estrelas em cada puzzle, sendo estas estrelas dependentes do seu desempenho na resolução do mesmo.

Cargo-Bot pretende familiarizar os jogadores com alguns conceitos de programação, tais como *loops* e condições. O objectivo de cada nível é apresentado na parte superior do ecrã e, para alcançar esse objectivo, o jogador deve construir um código que faça o guindaste movimentar-se de forma a colocar as caixas no sítio desejado (figura 16).

A interface é composta por quatro linhas de comandos (três linhas com oito *slots* e a última linha só com cinco *slots*) e um botão *play* na parte inferior da aplicação para executar os comandos colocados nas linhas. À direita, encontra-se uma *toolbox* que contém um conjunto de setas utilizadas para movimentar o guindaste (esquerda, direita e baixo), quatro números, correspondendo cada um a uma linha, e um botão para limpar todos os *slots* presentes nas linhas (figura 17). Existe também o botão *Hints*, que permite obter dicas para a resolução dos desafios.

Figura 16. Ambiente *Cargo-Bot* [50]

Figura 17. Construção de regras para alcançar o objectivo [51]

2.1.7. RoboZZle

RoboZZle é um jogo online, composto por vários puzzles, que desafia o jogador a programar um avião com o objectivo de apanhar todas as estrelas existentes no campo de jogo (figura 18) [15]. A mecânica do jogo é muito simple e permite uma ampla variedade de desafios, exigindo ao jogador que recorra à lógica para os resolver.

O jogador resolve esses desafios utilizando símbolos e cores para criar uma sequência de movimentos que permitem ao avião apanhar todas as estrelas. Esses símbolos e cores são dispostos em *slots,* apresentados no canto inferior direito do ecrã, que definem as regras para o avião (figura 19). Por exemplo, se o jogador colocar num slot uma seta para a direita, juntamente com a cor vermelha, definirá a regra de que o avião deve virar à direita caso encontre uma área a vermelho.

Após criar todas as regras, o jogador selecciona o botão *Go!* e observa o caminho que o avião percorre de acordo com as regras criadas anteriormente. Para além de os resolver, o jogador pode também criar os seus próprios puzzles e colocá-los online de forma a que os outros jogadores os possam jogar.

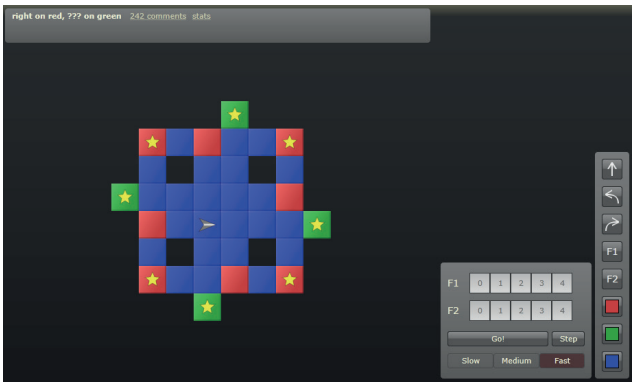


Figura 18. Ambiente *RoboZZle* [52]

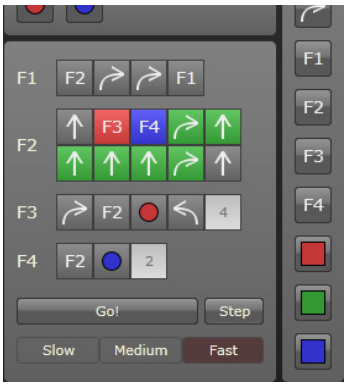


Figura 19. Construção de regras [53]

2.1.8. Help Gidget

Help Gidget é um jogo online, desenvolvido para ajudar na aprendizagem de conceitos de programação através de puzzles (figura 20). Este jogo mantém os jogadores interessados, fornecendo uma história interessante, com *feedback* imediato, e ferramentas para ajudar os jogadores quando estes estão com dificuldades [16].

Este jogo é composto por vários puzzles, sendo estes resolvidos através da utilização de código simples. A dificuldade do jogo vai aumentando progressivamente, à medida que o jogador vai passando os níveis. Por cada conjunto de 5, 6 ou 7 níveis, o jogador é introduzido num novo conceito de programação.

Possui uma abordagem diferente da utilizada nas tecnologias de aprendizagem de programação já existentes. Ao invés de tentar motivar as crianças através da criatividade (como no Scratch), ou fornecer instruções através de tutoriais (como no CodeAcademy), Help Gidget tenta traduzir a programação em pequenos puzzles para o jogador treinar a lógica e a resolução de problemas [17].

A interface é composta por quatro áreas distintas (figura 21). A primeira área, no canto superior esquerdo, é o local onde o jogador escreve o código. Abaixo, encontra-se a área referente ao objectivo a alcançar. Este objectivo é apresentado em forma de código, o que pode tornar-se confuso e incompreensível para quem não tem familiaridade com uma linguagem de programação. No centro, encontra-se a área onde estão contidos os personagens e objectos que interagem entre si, formando o puzzle. Por fim, no lado direito, encontra-se alguma informação sobre o personagem principal.

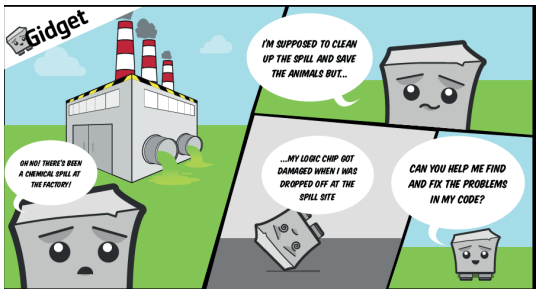


Figura 20. *Help Gidget* [54]

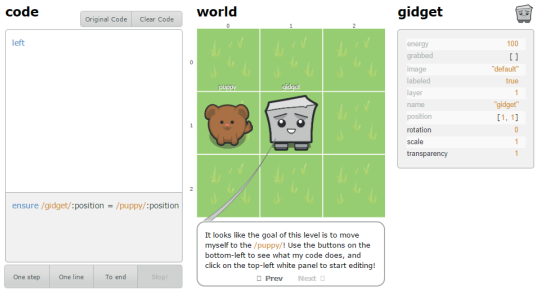


Figura 21. Ambiente *Help Gidget* [54]

Síntese dos diferentes ambientes de programação referidos anteriormente.

Ambiente	Elementos de Programação	Modo de Programação	Características
Logo	Introduz todos os elementos de programação	Código escrito	Utilização de um personagem para animar; Código simplificado para mais fácil aprendizagem; O jogador tem liberdade total para explorar e desenhar utilizando código.
Scratch	Introduz todos os elementos de programação	Blocos com instruções	Utilização de personagens e texto para animar; Utilização de blocos para facilitar a construção de código e eliminar possíveis erros de síntese; Feedback imediato após construção de código; O jogador tem liberdade total para explorar e construir código utilizando os blocos de instruções existentes.
Minecraft: Hour of Code	Funções, Iterações, Condições	Blocos com instruções	Utilização dos personagens do jogo Minecraft para motivar a programação nas crianças; Possui níveis com objectivos definidos, e um nível final para que o jogador possa explorar à sua vontade; Utilização de blocos para facilitar a construção de código e eliminar possíveis erros de síntese; Feedback imediato após construção de código, destacando a linha de código que corresponde à acção que está a ser executada.
Daisy The Dinossaur	Funções, Iterações, Condições	Blocos com instruções	Utilização de um personagem para animar; Possui níveis com objetivos definidos, e um nível final para que o jogador possa explorar à sua vontade; Utilização de blocos para facilitar construção de código e eliminar possíveis erros de síntese; Feedback imediato após construção de código, destacando a linha de código que corresponde à acção que está a ser executada.
Kodu	Funções, Variáveis, Condições	Point and Click	Utilização de personagens para animar; Não possui objectivos definidos; Utilização de acções predefinidas para animar os personagens colocados no mundo criado.

Ambiente	Elementos de Programação	Modo de Programação	Características
Cargo-Bot	Funções, Recursividade	Símbolos	Utilização de símbolos para facilitar a construção de código; Cada nível possui um objetivo; Feedback imediato após construção de código, destacando o símbolo que corresponde à acção que está a ser executada.
RoboZZle	Funções, Condições, Iterações, Recursividade	Símbolos e cores	Utilização de símbolos para facilitar a construção de código; Cada nível possui um objetivo; Feedback imediato após construção de código, destacando o símbolo que corresponde à acção que está a ser executada.
Help Gidget	Introduz todos os elementos de programação	Código escrito	Utilização de um personagem para animar; Utilização de uma linguagem própria simplificada; Aprendizagem através de alteração de código fornecido; Cada puzzle possui um objetivo definido.

Tabela 1. Comparação de ambientes para a aprendizagem da programação

Analisando os diversos ambientes já desenvolvidos, observamos que existem vários métodos de construção de código com o intuito de simplificar a aprendizagem da programação e de conseguir que o jogador consiga construir uma solução para o problema que lhe é colocado sem que precise de aprender uma sintaxe complexa de programação.

Também é possível verificar que, nos diversos ambientes analisados, são introduzidos alguns elementos da programação, tais como, variáveis, funções, condições e recursividade, o que permite ao jogador começar a desenvolver a lógica da programação.

Por fim, é de realçar que a maioria dos ambientes possui *feedback* imediato, o que permite ao jogador identificar possíveis erros no código que construiu, facilitando a aprendizagem e o desenvolvimento da lógica e da resolução de problemas.

2.2. Desenvolvimento Cognitivo das Crianças

Ao longo do crescimento, a criança atravessa vários estágios de desenvolvimento, adquirindo novas competências em cada um deles. Estas competências são resultado da assimilação e acomodação de experiências vividas pela criança.

Várias foram as teorias desenvolvidas sobre o desenvolvimento cognitivo das crianças ao longo do tempo. Uma delas é a do filósofo Jean Piaget, que teve um considerável impacto no campo da ciência da computação devido a Seymour Papert ter utilizado o seu trabalho como fundamento no desenvolvimento da linguagem de programação Logo, e de Alan Kay usar as suas teorias como base para o sistema de programação Dynabook, um computador pessoal para crianças de todas as idades [55].

2.2.1. Jean Piaget

Piaget [18], afirma que a adaptação é fundamental para o funcionamento biológico, assim como para o funcionamento cognitivo. Ele considera que o processo de construção do conhecimento é iniciado com um desequilíbrio, provocado pela vivência de novas experiências. Para manter este equilíbrio são utilizados dois processos: assimilação e acomodação [19].

O processo de assimilação é caracterizado pelo processo cognitivo de classificação e incorporação de novas experiências; e o processo de acomodação caracteriza-se pela modificação ou criação de estruturas cognitivas após vivência de novas experiências.

Estes dois processos possibilitam a adaptação da criança ao meio que a rodeia, potencializando a aprendizagem e desenvolvimento de competências durante o seu crescimento.

2.2.1.1. Estágios do Conhecimento

Segundo Piaget, existem quatro estágios no crescimento da criança. Em cada um deles a criança desenvolve novas competências e conhecimentos, ajustando-se assim, ao meio em que vive de forma a conseguir ultrapassar os obstáculos com os quais se vai defrontando.

Estágio 1: Sensoriomotor Thought (0 – 2 anos)

Neste estágio, o conhecimento do mundo é feito através dos sentidos (visão, olfacto, paladar, tacto e audição) e das acções motoras da criança sobre o mesmo (agarrar, morder, tocar, etc). Neste estágio, o pensamento representacional vai sendo desenvolvido à medida que a criança começa a ser capaz de representar, mentalmente, objectos, eventos e conceitos.

Estágio 2: Preoperational Thought (2 – 7 anos)

Neste estágio, é desenvolvida a capacidade de representar mentalmente objectos, eventos e conceitos, assim como o pensamento intuitivo (raciocínio) com base na experiência pessoal. No entanto, neste estágio a criança ainda não tem capacidade de visualizar a perspectiva do outro (egocentrismo).

O aspecto mais visível durante o desenvolvimento do pensamento pré-operatório é a fala, que neste estágio sofre um grande desenvolvimento, pois é neste período que a criança desenvolve a linguagem.

Estágio 3: Concrete Operational Thought (7 – 11 anos)

Neste estágio, é desenvolvido o pensamento descentralizado (capacidade de considerar diferentes aspectos de um problema) e a capacidade de utilizar o pensamento lógico para resolver problemas concretos. Nesta fase, a criança ainda não consegue utilizar operações lógicas em conceitos, objectos ou contextos abstratos.

Estágio 4: Formal Operational Thought (12 + anos)

Neste estágio, o desenvolvimento cognitivo adquire todo o seu potencial e é desenvolvida a capacidade hipotético-dedutiva, fazendo com que seja capaz de entender conceitos abstratos.

É nesta fase que a criança desenvolve a sua própria identidade, podendo haver, neste período, problemas existenciais e dúvidas entre o certo e o errado.

Cada um destes estágios é caracterizado por diferentes formas de organização mental que possibilitam diferentes maneiras do indivíduo se relacionar com a realidade que o rodeia. De forma geral, todos os indivíduos vivenciam estes 4 estágio, porém, o início e o fim de cada um pode sofrer alterações devido às características da estrutura biológica de cada um e da riqueza dos estímulos proporcionados pelo meio ambiente em que estiver inserido [56].

Jean Piaget não foi o único a desenvolver uma teoria para o desenvolvimento cognitivo das crianças. Jerome Bruner também se debruçou sobre este tema, e Alan Kay foi influenciado por Bruner ao perceber que podia utilizar a sua teoria no desenvolvimento de interfaces de computadores, a fim de aumentar a capacidade de aprendizagem em utilizadores de todas as idades.

2.2.2. Jerome Bruner

Para Bruner, o desenvolvimento cognitivo envolve a interacção entre as capacidades humanas básicas e as tecnologias culturalmente inventadas. Estas tecnologias incluem computadores e telemóveis, mas também noções mais abstractas, como o modo de uma cultura categorizar os fenómenos e a própria linguagem [57].

Os resultados do crescimento cognitivo não devem incluir apenas conceitos, categorias e procedimentos criados e incutidos pela cultura, mas também a capacidade de criação de novos.

Desta forma, Bruner defendia uma participação activa do aluno no processo de aprendizagem, propondo o conceito de aprendizagem por descoberta [58]. Do seu ponto de vista, os professores devem criar condições para que as crianças, ao explorarem e tentarem resolver os problemas, aprendam e incorporem significativamente esse conhecimento na sua estrutura cognitiva.

2.2.1.1. Estágios do Conhecimento

À semelhanca de Piaget, Jerome Bruner procurou tipificar o desenvolvimento cognitivo num conjunto de estágios, tendo resultado em três estágios: Enativo, Icónico, Simbólico.

Estágio 1: Enactive (0 – 3 anos)

Neste estágio, a representação dos acontecimentos passados é feita através de respostas motoras apropriadas, privilegiando a acção como forma de representação do que é real. A criança desta faixa etária aprende, sobretudo, através da manipulação de objectos. Neste estágio, a criança age com base em reflexos, simples e condicionados.

Estágio 2: Iconic (3 – 9 anos)

Neste estágio, a representação baseia-se na organização visual, no uso de imagens sinópticas e na organização de percepções e imagens. A criança torna-se capaz de reproduzir imagens, ainda sem transposição. Neste estágio, a sua memória visual torna-se concreta e específica.

Estágio 3: Symbolic (10 + anos)

Neste estágio, a criança torna-se capaz de representar a realidade através de uma linguagem simbólica, de carácter abstracto e sem dependência directa da realidade. O conhecimento é caracterizado, principalmente, por palavras, símbolos matemáticos ou outros sistemas de símbolos.

A teoria do desenvolvimento cognitivo de Bruner é distinta de outras teorias baseadas em estágios do desenvolvimento cognitivo, uma vez que este propõe que mesmo as crianças mais jovens podem aprender conceitos difíceis tendo apoio pedagógico apropriado [58].

Embora Jerome Bruner, à semelhança de Jean Piaget, coloque a maturação e a interacção do sujeito com o ambiente no centro do processo de desenvolvimento e da formação da criança, este acentua o carácter contextual dos factos psicológicos. A abertura à influência do contexto e do ambente social no processo de desenvolvimento e da formação torna a teoria de Jerome Bruner mais abrangente do que a teoria de Jean Piaget, fazendo com que consiga incorporar a transmissão social, o processo de identificação e a imitação no processo de desenvolvimento e formação.

A teoria de Bruner incorpora, de uma forma coerente, quer as contribuições do maturacionismo quer os contributos do ambientalismo, pois é através de uns e de outros que a criança organiza os diferentes modos de representação da realidade, utilizando as técnicas que a sua cultura lhe transmite [59].

2.3. Aprendizagem Baseada em Problemas (Problem-Based)

Segundo a teoria de Bruner, referida no sub-capítulo anterior, as crianças exploram e tentam resolver os problemas que lhes são colocados, aprendendo e incorporando significativamente esses conhecimentos na sua estrutura cognitiva.

A aprendizagem baseada em problemas é um método de ensino centrado no aluno, onde este aprende sobre uma dada matéria através da experiência obtida após a resolução de um problema referente a essa mesma matéria. Este método de ensino procura uma estratégia formativa que leve os alunos a serem confrontados com problemas contextualizados e pouco estruturados, de modo a terem de encontrar possíveis soluções. Desta forma, os estudantes aprendem a determinar o que é verdadeiramente necessário saber para resolver os problemas [23].

Sendo também um método de aprendizagem que ocorre muitas vezes em grupos, a aprendizagem baseada em problemas permite desenvolver o pensamento crítico dos alunos e construir, em conjunto, soluções mais criativas e inovadoras. Este método promove também o desenvolvimento do raciocínio lógico, da comunicação e da colaboração em grupo.

A aprendizagem baseada em problemas foi, nos anos 50, desenvolvida para o ensino da medicina. No entanto, este tipo de aprendizagem já é utilizada noutras áreas de ensino, tais como, Matemática, Direito, Educação, Economia e Engenharia. Barrows [23], afirma que a aprendizagem através deste método de ensino é muito mais efetiva na criação de conhecimento útil para o futuro do que o método tradicional de ensino, baseado na memória.

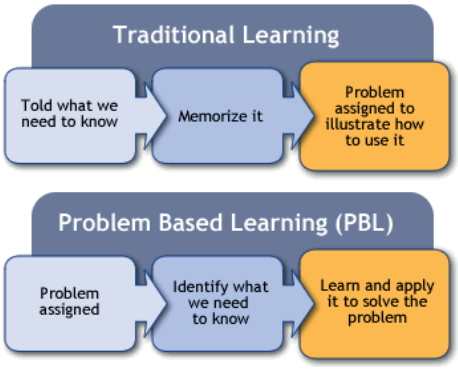


Figura 22. Comparação entre o método de aprendizagem tradicional e o método de aprendizagem baseada em problemas [38]

A aprendizagem baseada em problemas potencia a motivação na aprendizagem; possibilita o estudo de matérias relevantes, a criação de pensamento crítico e criativo, e a participação e decisão sobre o processo de aprendizagem; permite o estudo de situações reais úteis para o futuro profissional; e cria capacidade de análise, decisão, delegação e trabalho em grupo [22].

Segundo Jerome Bruner, as crianças tendem a precisar da ajuda de professores ou de tutores quando começam a aprender novos conceitos. No entanto, à medida que estes vão sendo consolidados, as crianças tornam-se mais independentes e acabam por deixar de precisar da ajuda inicial [62].

Este apoio dado durante o processo de aprendizagem é designado de *Scaffolding*. Brunner acreditava que os jogos são um exemplo de *Scaffolding*, uma vez que proporcionam uma aprendizagem e participação activa através da execução de tarefas [62].

2.4. Play e Game

Os termos *play* e *game* são muitas vezes utilizados como se fossem o mesmo, o que não é verdade. *Games* são formas de *play,* e *play* é uma actividade desejada e apreciada pelo próprio indivíduo, que desfruta do momento em que pratica essa atividade. *Game* é objectivo e construído de acordo com regras, enquanto que *play* é subjectivo e fundamentado no jogador.

Play

Play é uma atividade ou ocupação voluntária, exercida dentro de certos e determinados limites de tempo e espaço, segundo regras livremente consentidas, mas absolutamente obrigatórias, dotado de um fim em si mesmo, acompanhado de um sentimento de tensão e alegria, e de uma consciência de ser diferente da vida quotidiana [25].

Huizinga argumenta que o termo *play* é mais antigo que a cultura, sendo esta definida numa sociedade humana. "Os animais brincam tal como os homens. Basta que observemos os cães para constatar que, durante o seu crescimento, encontram-se presentes todos os elementos essenciais de *play*. Convidam-se uns aos outros para brincar mediante um certo ritual de atitudes e gestos. Respeitam a regra que os proíbe de morderem-se uns aos outros, ou pelo menos com violência. Fingem ficar zangados e, o mais importante, experimentam o prazer e o divertimento." [26].

Tal como os animais, também os humanos utilizam formas de play para desenvolverem novas capacidades durante o seu crescimento. Desta forma é incentivada a comunicação, a descoberta do mundo, a socialização e o desenvolvimento intelectual. É através do acto de brincar que a criança é confrontada com novos desafios e emoções, e, ao conseguir ultrapassar esses desafios, a criança adquire novos conhecimentos e aumenta as suas capacidades emocionaise intelectuais.

Game

Game é uma actividade que envolve um ou mais jogadores, que assumem papéis enquanto tentam alcançar um objectivo. As regras, definidas no jogo, determinam o que os jogadores podem fazer: quais as acções que podem executar, quais as restrições a que estão sujeitos, que recursos são disponibilizados, e como as acções que executam influenciam o espaço de jogo [26].

Através dos jogos, o jogador consegue adquirir e desenvolver competências emocionais e intelectuais, ao mesmo tempo que se diverte. É possível transmitir informação detalhada e complexa de uma forma divertida e motivante. Jogar dá prazer ao jogador, estando esse prazer ligado ao controlo, à acção e ao significado do jogo sobre o jogador.

Os jogos são constituídos por três componentes [27]:

Jogadores, que são aqueles que participam no jogo;

Regras, que definem os limites do jogo;

Objectivos, que são metas que o jogador deve alcançar durante o jogo.

Nos videojogos, por exemplo, os jogadores colocam-se no lugar de um personagem que se encontra num mundo virtual e que possui um objectivo que deve ser alcançado. Este personagem possui uma mente e corpo virtuais, isto é, existe e actua sobre o mundo do videojogo mas não existe na realidade, tornando-se na mente e corpo do jogador durante o período de tempo em que este está a jogar [27].

Ao colocar-se no lugar do personagem, o jogador consegue sentir o que ocorre em redor deste como se fosse real. O ambiente, o som e os desafios são algumas das características que permitem ao jogador sentir prazer enquanto joga um videojogo [27].

Actualmente, existem diversos géneros de videojogos, sendo estes categorizados relativamente ao tipo de *gameplay* que é oferecido ao jogador. Entre os vários géneros, existem os serious games: videojogos que têm como objectivo transmitir conhecimento. Videojogos deste género foram analisados no sub-capítulo 2.1.

2.5. Serious Games

Existem, nos dias de hoje, tecnologias que permitem transmitir conhecimento de forma simples e divertida. Através dos videojogos é possível transmitir informação detalhada e complexa de uma forma mais divertida e envolvente.

Serious games são videojogos desenvolvidos com propósito educacional, não estando focados, essencialmente, no entretenimento. Porém, como qualquer jogo, dependem de regras, mecânicas e elementos de jogos, tais como, enredo, pontuação, missões e interacção entre personagens [31].

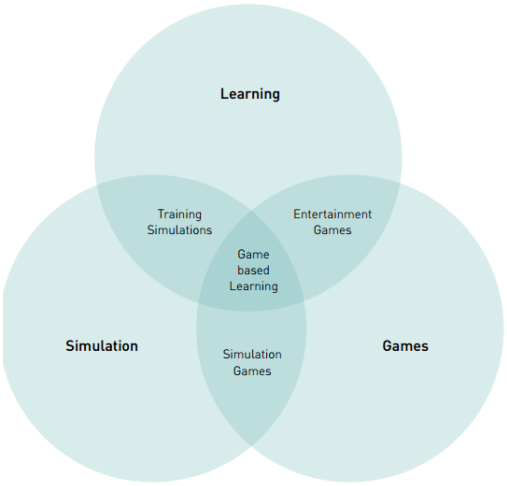


Figura 23. Interligação entre jogos, simulação e aprendizagem [30]

Normalmente, este tipo de videojogos são utilizados para treinar, publicitar, simular ou educar [32]. Assentes no método de aprendizagem baseada em problemas, os serious games são bons modelos de ensino para formação profissional e vocacional, que implicam aprender os procedimentos e o modo de atuar em cada situação [30]. Existem diversas áreas de formação que utilizam serious games, tais como, a militar, a da saúde e a governamental.

Os serviços militares foram os primeiros a tirar proveito do uso de simuladores e de videojogos, quer como recurso de formação, de terapia ou mesmo como forma para angariar novos recrutas [33]. O primeiro serious game, desenhado e utilizado, para treino militar foi Army Battlezone, em 1980 [32]. Posteriormente, em 2002, foi lançado o jogo America's Army que é considerado uma das maiores fontes de recrutamento militar nos Estados Unidos da América [33].

Relativamente à área da saúde, este tipo de jogos estão a aumentar cada vez mais, uma vez que o interesse pela saúde e pelo aspecto físico está cada vez mais assente na sociedade.

Existem videojogos de fitness, tais como Dance Dance Revolution, que motivam o jogador a fazer exercício através movimentos simples que são apresentados no ecrã; e videojogos que ajudam a ter uma melhor alimentação, como por exemplo, Hungry Red Planet, que transmite às crianças alguns conhecimentos de nutrição e como ter hábitos de alimentação saudável.

Além dos exemplos anteriormente referidos, existem serious games nesta área que são direcionados para a aprendizagem da medicina. Por exemplo, existem jogos de treino e simulação, que permitem, a um estudante de medicina, treinar o modo como deve proceder numa cirurgia [32].

Na área governamental, os videojogos podem dizer respeito a vários tipos de tarefas e situações, como por exemplo, tipos de gestão de problemas, como lidar com ataques terroristas, surtos de doenças, riscos biológicos, questões de política, urbanismo, controlo de tráfego, combate a incêndios, equilíbrio orçamental e condução defensiva [32].

	Serious Games	**Videojogos para entretenimento**
Objectivo	Resolução de problemas	Experiência enriquecedora
Foco	Aprendizagem	Divertimento
Simulações	Premissas necessárias para simulações funcionais	Processos de simulação simples
Comunicação	Deve reflectir uma cominicação natural (i.e. não perfeita)	Comunicação frequentemente perfeita

Tabela 2. Diferença entre serious games e videojogos para entretenimento

3 - Proposta de Design

Este capítulo é referente à primeira fase do desenvolvimento do projecto, sendo definidos o conceito e as mecânicas do jogo, nomeadamente, o modo como o jogador interage com a aplicação, quais as regras dessa interacção e a qual a história do jogo.

Depois de analisadas as ferramentas de aprendizagem apresentadas no sub-capítulo 2.1, foi possível perceber quais as características presentes nessas ferramentas que facilitam a aprendizagem e que devem estar contidas neste projecto, assim como aquelas que não proporcionam a motivação necessária para manter o interesse do jogador e que devem ser alteradas ou removidas.

Relativamente ao método de construção do código, foi estudada e analisada uma hipótese colocada que simplifica a forma como é construído o código, mas que, ao mesmo tempo, é bastante versátil e com a qual se pode construir algo complexo.

Por fim, foi desenvolvida a história que servirá como base aos puzzles existentes nos diversos níveis. Uma história simples, mas que posibilita a construção de puzzles simples ou complexos, de acordo com o evoluir da mesma.

3.1. Conceito do Jogo

Após pesquisa e análise das ferramentas de aprendizagem apresentadas no sub-capítulo 2.1, foi necessário definir alguns requisitos que o jogo deveria satisfazer, de forma a tornar-se numa boa ferramenta de aprendizagem.

Foram definidos os seguintes requisitos:

- Ter um método de construção de código simples e fácil de aprender;
- Ter um objectivo definido para cada nível;
- Aumentar, progressivamente, a dificuldade;
- Introduzir elementos de programação, tais como variáveis e condições;
- Incluir *feedback* visual;
- Incluir *feedback* sonoro.

Relativamente ao primeiro requisito, foi colocada a hipótese de utilizar Petri Nets como método de construção de código devido à sua simplicidade e facilidade de leitura e construção. Após o estudo e análise do funcionamento e contrução das mesmas concluiu-se que, para que fosse possível utilizá-las para este fim, estas teriam de sofrer algumas alterações no seu design, nomeadamente, a introdução de imagens e de cor. No entanto, as regras e o modo de construção das mesmas manter-se-iam iguais.

O objectivo do segundo e terceiro requisitos é manter o jogador motivado e interessado ao longo dos diferentes níveis. Ao ser definido um objectivo para cada nível, o jogador acaba por sentir uma maior motivação para continuar a jogar até conseguir atingi-lo. Por outro lado, aumentando progressivamente a dificuldade dos níveis, o jogador vai começar a sentir uma evolução relativamente aos seus conhecimentos, estimulando-o a continuar a jogar.

Por fim, a introdução de elementos de programação, como variáveis e condições, promove o desenvolvimento da lógica da programação; e a introdução de *feedback* visual e sonoro permite ao jogador detectar mais facilmente possíveis erros no código construído e aperceber-se de acontecimentos que ocorrem ao longo do nível.

3.1.1. Método de Construção da Solução

Tendo sido colocada a hipótese de utilizar Petri Nets na aplicação como representação manipulável da solução que o jogador construirá para o problema colocado, foi necessário perceber como é que estas funcionam, quais as regras utilizadas para a sua construção e de que modo poderiam ser utilizadas no contexto da aplicação. Esta hipótese teve origem em ensaios anteriores do uso de Petri Nets com alunos em *game design*, havendo uma publicação (Araujo e Roque 2009) [35] sobre o assunto, que constituiu uma base de trabalho.

No sub-capítulo seguinte, é descrito o que é uma Petri Net, qual o propósito da sua utilização e quais as regras que possui e que devem ser cumpridas durante a sua construção.

3.1.1.1. Petri Net

Petri Net é uma ferramenta gráfica utilizada para a descrição e análise de processos que surgem em sistemas complexos. Esta ferramenta, juntamente com as regras que a definem, foi inventada em 1939 pelo alemão Carl Adam Petri, com o objectivo de descrever processos químicos [34].

As Petri Nets são utilizadas para a modelação de sistemas, técnicas de análise e representações gráficas [35]. Graficamente, as Petri Nets são representadas num diagrama composto por círculos, que representam os estados; por barras ou quadrados, que representam as transições de estados; e por arcos, que fazem a ligação entre os estados e as transições. Os estados podem representar condições, dados de *input/output* ou recursos; enquanto que as transições podem ser interpretadas como eventos, tarefas, entre outros [36].

Um estado pode conter um ou mais *tokens*, sendo estes caracterizados por um ponto. Podem representar recursos ou se uma dada condição é verdadeira ou falsa. Os *tokens* são movidos, através das transições, de um estado para outro. Quando a transição é accionada, esta remove um *token* dos estados de entrada, e acrescenta um token nos estados de saída (figuras 24 e 25) [36].

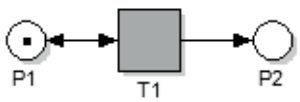


Figura 24. Petri net simples composta por dois estados e uma transição

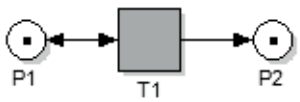


Figura 25. Resultado após transição ser accionada

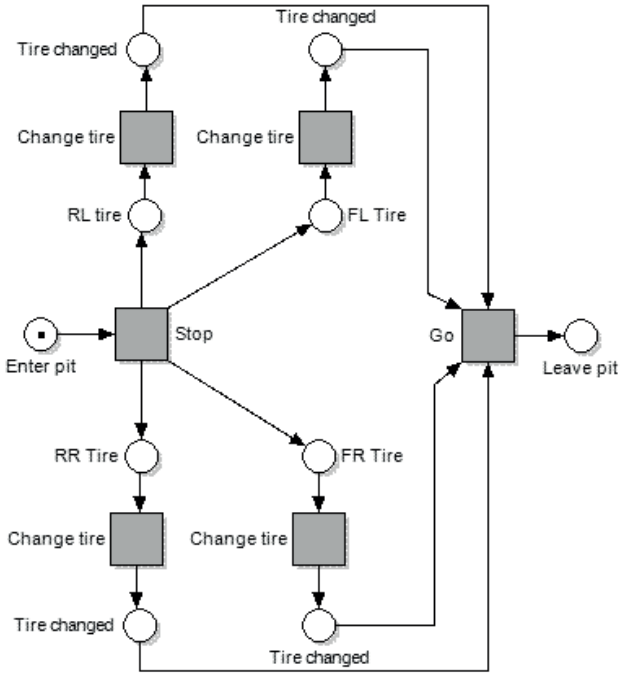


Figura 26. Exemplo de uma Petri net representativa da mudança de quatro pneus

Observado a figura 26, é possível constatar a existência um *token* no estado Enter pit. Este *token* vai ser eliminado e vão ser adicionados *tokens* nos estados RL Tire, FL Tire, RR Tire e FR Tire, através da transição Stop. À medida que cada pneu vai sendo substituído, o *token* correspondente vai passando para o estado seguinte até alcançar o estado Tire changed. A transição Go só é accionada quando todos estados ligados a esta tiverem um *token*, ou seja, quando todos os pneus estiverem substituídos [36].

Seguindo a lógica de construção da rede do exemplo da figura 27, e tendo em conta o modo de construção das Petri Nets, são definidas algumas regras que devem ser seguidas aquando da construção da aplicação:

- As variáveis/objectos presentes no jogo são comparados a locais;
- As acções possíveis de realizar são comparadas a transições;
- Só é possível ligar variáveis (locais) a acções;
- Só é possível ligar acções a variáveis;
- Uma variável pode depender de uma ou mais acções;
- Uma acção pode produzir ou alterar *tokens* numa ou mais variáveis (locais).

3.1.2. Descrição do Jogo

"Duarte, um jovem rapaz, decide ir passear com o seu cão, Blu. Durante o passeio, Blu consegue soltar-se da trela e começa a fugir. Duarte não o consegue apanhar, mas apercebe-se de que o seu cão entrou dentro de um edifício abandonado e decide ir atrás dele para o encontrar."

É nesta parte da história que começa o jogo.

Controlando o Duarte, o jogador deve encontrar a solução para cada puzzle de forma a ir avançando para os seguintes níveis e, no final, conseguir encontrar o Blu. O jogador controla o personagem através da construção de uma rede de acções, definidas de acordo com o local ou obstáculo com o qual o Duarte se depara a dada altura do nível.

Em cada nível, é pedido ao jogador que defina, seguindo as regras das Petri Nets, as acções para cada um dos personagens presentes no ambiente de jogo. Estes personagens serão necessários para resolver o problema colocado tendo, cada um, acções próprias que permitem ultrapassar os obstáculos. Os níveis vão aumentando gradualmente de dificuldade, desafiando o jogador a construir regras cada vez mais complexas.

De modo a ser mais fácil aprender a interagir com o ecrã de jogo e a construir as regras, o nível 1 dá indicações ao jogador de como deve proceder para resolver o problema colocado através de animações e notificações.

O nível 2 permite ao jogador consolidar a aprendizagem do nível anterior, sendo colocado um desafio semelhante ao do nível 1.

No nível 3 é introduzido um novo elemento, o contador, que o jogador deve utilizar para conseguir construir a solução correcta e concluir o nível com sucesso.

No nível 4 é introduzido um novo agente, o fantasma. Para além de construir regras para o Duarte, o jogador deverá também construir regras para o fantasma que, neste nível, ajuda o Duarte a superar o desafio colocado. Ao ser introduzido um novo agente, é fornecida informação ao jogador sobre o mesmo (através de notificações) para que este não se sinta confuso e sem saber o que fazer.

Em todos os níveis, após a construção da solução, é possível visualizar o resultado das regras construídas através da animação do personagem. Este vai realizando as acções definidas pelo jogador, ao mesmo tempo que é destacada a regra que corresponde à acção que está a ser executada. No final da animação, caso a solução construída não seja a correcta, a área de construção volta a estar a branco e o jogador pode construir uma nova solução.

O objectivo de cada nível é sempre o mesmo: conseguir chegar à porta que dá acesso à próxima sala. Ao conseguir chegar à porta, e sair daquela sala, o jogador desbloqueia um novo nível.

3.1.3. Modo de Jogar

Ao iniciar o primeiro nível, o jogador recebe informações sobre o modo de interagir com o ecrã de jogo. Através de animações, é explicado ao jogador que este deve arrastar as acções para o centro da área de construção e, de seguida, unir os objectos às acções.

Após o jogador realizar as acções anteriores, são lhe explicadas, através de um painel de notificação, as regras que acabou de criar e que solucionam o problema. Desta forma, o jogador aprende como se lêem as regras e torna-se capaz de criar regras para os desafios que se seguem. Neste nível é também explicado ao jogador que, ao terminar de criar as regras, deve seleccionar o botão Play caso pretenda visualizar o resultado das mesmas, ou seleccionar o botão Stop caso pretenda alterá-las. Assim que o jogador selecciona o botão Play, a animação do Duarte é executada e o nível é concluído com sucesso.

No segundo nível, o jogador deve executar as acções aprendidas no nível anterior, podendo construir e alterar as regras como desejar. Uma vez que são utilizadas acções e objectos semelhantes aos do nível 1, o jogador acaba por se sentir capaz de resolver o problema e consolida, desta forma, toda a informação transmitida anteriormente. Neste segundo nível já não é fornecida ajuda ao jogador e este acaba por ter de construir as regras de acordo com as conclusões que retira após visualização da animação do Duarte. É a partir deste nível que se pede ao jogador que utilize o raciocíno lógico para conseguir encontrar a solução.

O terceiro nível introduz um novo elemento, o contador, que o jogador deverá utilizar na construção das regras. Este novo elemento é introduzido ao jogador através de uma notificação que aparece no ecrã no início do nível, e o jogador deverá, utilizando o raciocínio lógico, perceber o que deve ser ligado ao contador de modo a construir a regra correcta. Assim, é pedido ao jogador que recorra à visualização da animação do Duarte e ao destaque das regras para raciocinar sobre o problema e perceber quais as regras que deve alterar caso a solução que tenha criado não seja a correcta.

O quarto nível introduz um novo personagem, o Fantasma, e pede ao jogador que crie regras tanto para o Duarte como para o Fantasma. A introdução de um novo personagem leva a que o jogador tenha de recorrer à lógica, de forma a criar uma solução que envolva os dois personagens para a resolução do problema. Tal como no terceiro nível, a introdução deste novo elemento é explicado ao jogador através de notificações que aparecem no ecrã no início do nível.

Com a introdução, no primeiro nível, das animações referentes às acções que o jogador deve executar; com a explicação do modo de ler as regras contruídas; com o problema do segundo nível, semelhante ao do primeiro de forma a facilitar a aprendizagem inicial; com a introdução de novos elementos e com o aumento gradual da dificuldade no terceiro e quarto níveis, pretende-se que esta aplicação contribua como *scaffolding* para o jogador aprender a raciocinar em termos de lógica de programação.

4 - Design da Aplicação

Este capítulo é referente à fase de desenvolvimento do design da aplicação, desde o protótipo de baixa fidelidade até aos modelos de alta fidelidade utilizados na aplicação.

Inicialmente, foram desenvolvidos protótipos de baixa fidelidade para os diferentes níveis que compõem a aplicação. Com estes protótipos foram realizados testes, com diferentes jogadores, de forma a perceber se a disposição dos diferentes elementos do ecrã de jogo era perceptível para o jogador.

Posteriormente, foram definidos os diferentes ecrãs que iriam compor a interface da aplicação, assim como a ligação entre eles. Depois de definidos os ecrãs, foi iniciado o desenvolvimento dos elementos que os compõem e definidas as cores da aplicação.

Por fim, com os elementos de identidade e cores definidos, foi realizado o desenvolvimento dos ecrãs de alta resolução que serão apresentados na aplicação funcional.

4.1. Protótipo de Baixa Fidelidade

Numa primeira fase, foram elaborados protótipos de baixa fidelidade do ecrã de jogo. Estes protótipos são referentes aos diferentes níveis do jogo, sendo o puzzle do primeiro nível pensado de forma a introduzir o jogador no modo de funcionamento do jogo.

O ecrã de jogo é composto por duas áreas distintas: a área onde é realizada a animação do personagem, e na qual é observado o resultado da solução construída; e a área de construção da solução. Estas duas complementam-se e, quando uma é activada, a outra torna-se inactiva.

Para além destas duas áreas, no ecrã estão contidos dois botões, Play e Stop. O primeiro deve ser seleccionado quando o jogador termina a construção das regras e pretende verificar qual o resultado obtido; o segundo é seleccionado quando o jogador pretende limpar a área de construção e construir uma nova solução. Em alguns níveis existe o botão Help que permite receber ajuda através de dicas.

Na elaboração do design foi tido em atenção que o jogo seria desenvolvido para tablet, tendo sido desenvolvida a interface de forma a que os objectos que a compõem fossem facilmente manipulados com os dedos. Por exemplo, os objectos não poderiam ser demasiado pequenos para que não houvesse o risco de não se conseguir seleccionar.

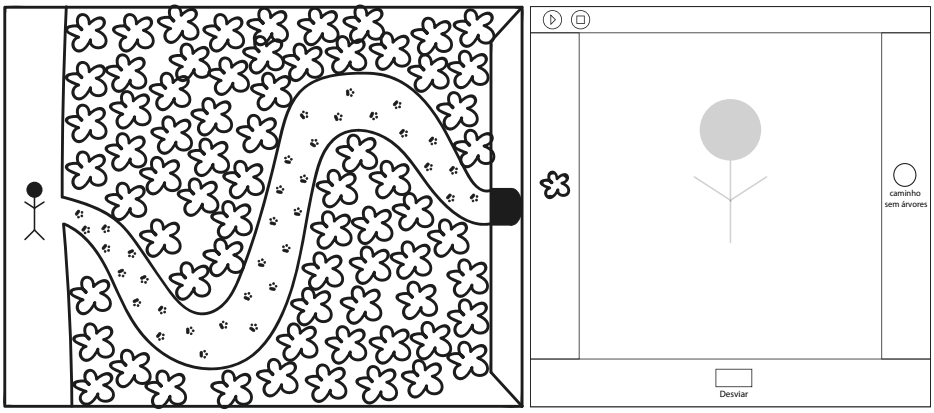


Figura 27. Protótipo de baixa fidelidade do ecrã de jogo - nível 1

Na figura 27, podemos observar a existência de uma área de animação do lado esquerdo e de uma área de construção da solução do lado direito. Os objectos e acções presentes na segunda área são necessários à construção de uma regra, sendo o objecto do lado esquerdo um local onde o Duarte possa estar; em baixo, a acção que o Duarte deve realizar; e o objecto do lado direito o local que o Duarte deve alcançar.

O primeiro protótipo foi realizado em papel, de forma a perceber se a disposição dos diferentes elementos presentes no ecrã era perceptível e fácil de utilizar por parte do jogador. Após a sua construção, este foi submetido a testes por parte de 5 utilizadores distintos.

Os utilizadores eram todos estudantes, com idades entre os 21 e os 27 anos, e apenas um tinha conhecimentos sobre Petri Nets. Durante os testes não foi dada qualquer tipo de informação acerca do funcionamento do protótipo ou do modo de construção das Petri Nets.

O objectivo destes testes era, não só receber *feedback* dos jogadores relativamente ao design, mas também relativamente aos puzzles construídos. Era necessário perceber se o jogador percebia o objectivo de cada nível, se conseguia interagir sem dificuldade com a aplicação e se conseguia perceber a lógica subjacente ao método de construção da solução.

4.1.1. Testes ao Protótipo em Papel

O que foi pedido a cada jogador submetido a estes testes é que interagisse com o papel como se de um tablet se tratasse. Podia tentar seleccionar o que quisesse, podendo essa selecção ser através de um toque, de um arrastar ou de qualquer outra forma que o jogador pudesse sentir que era a correcta.

Esta liberdade na interacção com o protótipo pretendia perceber de que forma é que o jogador se sente em relação à interface, e como, por instinto, lhe parece certa a interacção com os objectos. Por exemplo, para unir dois objectos na área de construção da solução, o jogador poderia seleccionar o primeiro, levantar o dedo, e seleccionar o segundo objecto, criando assim uma seta entre eles; ou seleccionar o primeiro, manter o dedo pressionado, e arrastá-lo até ao segundo objecto.

Ao todo, foram quatro os níveis que os jogadores testaram.

O primeiro nível (figuras 28 e 29) permite ao jogador perceber o que deve e pode fazer na aplicação. Através da apresentação de textos e de animações, o jogador é levado a executar determinadas acções, percebendo, posteriormente, o porquê de as executar através da visualização dos resultados obtidos.

O segundo nível permite ao jogador repetir o processo executado no primeiro. No entanto, são alterados alguns objectos (estados) e retiradas as animações que demonstram ao jogador o que deve fazer.

O terceiro e quarto níveis introduzem novos elementos, tornando-se mais desafiantes e complexos. Nestes níveis, o jogador já sabe como deve interagir com a aplicação e tenta resolver os puzzles utilizando a lógica.

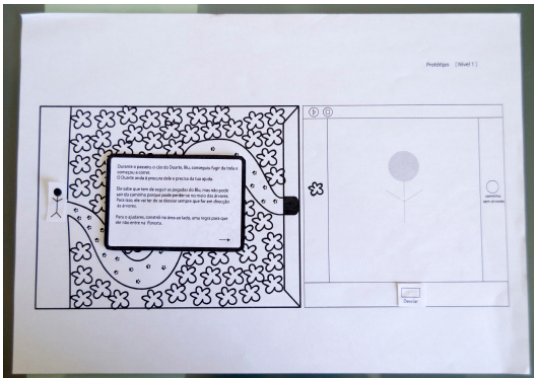


Figura 28. Protótipo em papel - Início do nível 1

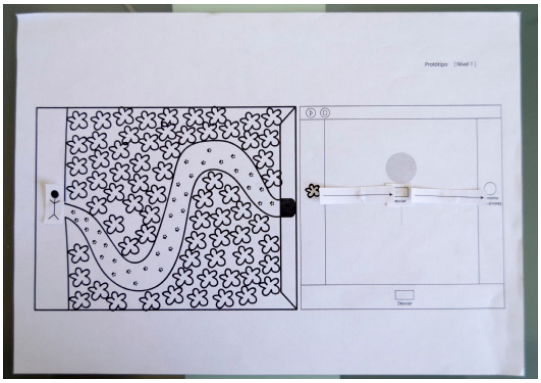


Figura 29. Protótipo em papel - Solução construída pelo jogador

Da observação da interacção dos jogadores com o protótipo foram retiradas as seguintes conclusões:

- Todos os jogadores sentiram facilidade em perceber o porquê da existência de duas áreas distintas no ecrã de jogo;
- Todos os jogadores perceberam rapidamente como interagir com a área de construção da solução após concluirem o primeiro nível;
- Para criar uma ligação entre um objecto e uma acção, na área de construção, todos os jogadores deslizaram o dedo entre os dois;
- Três dos cinco jogadores demonstraram alguma dificuldade na construção da solução do terceiro nível. Neste nível são disponibilizados mais objectos e acções do que nos dois primeiros;
- No terceiro e quarto níveis, mesmo existindo um botão de ajuda, quatro jogadores não clicaram porque sentiram vontade de conseguir encontrar a solução sozinhos;
- Dois jogadores criaram mais do que uma ligação entre uma variável e uma acção, o qual não será possível ser realizado. Este problema foi demonstrado no nível 3, quando o jogador percebe que tem de assustar três fantasmas.
- Todos os jogadores sentiram alguma confusão ao constatar que os locais finais não eram definido por um objecto/imagem, tal como acontece com os locais iniciais.

4.2. Interface da Aplicação

Após serem realizados os testes ao protótipo em papel, e de analisar os resultados obtidos, foram efectuadas algumas alterações ao ecrã de jogo. Para além disso, foi posteriormente pensada a estrutura dos ecrãs que compõem a aplicação e construído um esquema que os interliga entre si (figura 30).

Relativamente ao ecrã de jogo, foi decidido que os locais finais seriam representados da mesma forma que os locais iniciais, isto é, através de imagens; seriam apresentadas algumas informações ao jogador no início do jogo para que este tomasse conhecimento delas e não fosse prejudicado por não clicar no botão de ajuda; e o modo de ligar um local a uma acção seria efectuado através do movimento de deslizar do dedo.

Olhando para a aplicação no geral, esta consiste num conjunto de quatro ecrãs. O primeiro ecrã é o menu inicial, que possui três botões, dois dos quais permitem ao jogador ser direccionado para outro ecrã da aplicação.

O primeiro botão, Jogar, redirecciona o jogador para o ecrã de níveis. Neste ecrã, o jogador pode seleccionar o nível que pretende jogar e ser, posteriormente, direccionado para o ecrã de jogo. O segundo botão, Como Jogar, direcciona o jogador para um ecrã que lhe apresenta o modo de interagir com o ecrã de jogo e explica, sucintamente, como funciona a criação das regras. Existe mais um botão no menu inicial mas este apenas permite ao jogador fechar a aplicação.

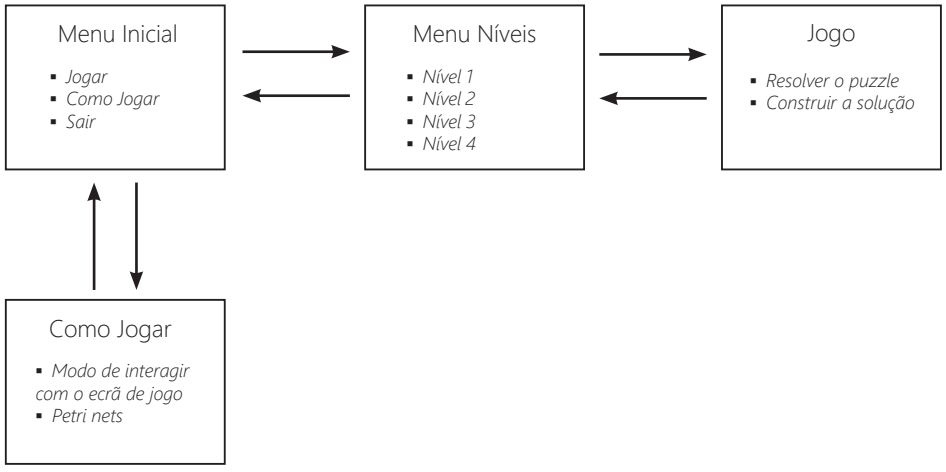


Figura 30. Ecrãs que compõem a aplicação

4.3. Elementos da Aplicação

Durante o desenvolvimento do protótipo em papel foram definidos vários elementos necessários ao correcto funcionamento da aplicação. Foram, durante esse período, definidos personagens, objectos, painéis de notificações, botões, entre outros.

Como já foi referido anteriormente, foram definidos alguns botões, necessários para a interacção e manipulação dos diferentes ecrãs, assim como para a sincronização das duas áreas existentes no ecrã de jogo. Referente ao ecrã de jogo, os elementos de cada nível foram definidos à medida que os níveis iam sendo criados.

É apresentado, de seguida, o objectivo do puzzle de cada nível. Foi através da construção de cada puzzle, e da solução para o mesmo, que se tornou possível definir os elementos necessários.

No primeiro nível, o Duarte deve seguir um caminho sem obstáculos de forma a chegar à porta da entrada da casa abandonada por onde o Blu entrou. Neste nível são necessários os seguintes elementos: o chão do caminho que ele deve seguir, os arbustos (obstáculos), a porta da casa abandonada e as acções.

No segundo nível, o Duarte deve percorrer a primeira sala de uma ponta à outra, tendo de assustar um fantasma para conseguir passar por ele e alcançar a porta que dá para a próxima sala. Novamente, são definidos os elementos necessários: o chão da sala, o fantasma, a porta para a outra sala e as acções.

No terceiro nível, o Duarte deve assustar não um, mas três fantasmas para conseguir obter uma chave e destrancar a porta que dá para a próxima sala. Os elementos necessários para este nível são o chão da sala, o fantasma, a chave, o cadeado da porta que está trancada, a porta e as acções.

No quarto nível, o Duarte, com a ajuda de um fantasma, deve conseguir apanhar um martelo e partir a parede que se encontra a meio da sala, de forma a conseguir alcançar a porta para a próxima sala. Os elementos necessários neste nível são o chão da sala, o fantasma, o Duarte, o martelo, a parede, a porta para a próxima sala e as acções.

Para além dos elementos referentes a cada nível e, uma vez que se trata de um jogo, é necessário o aparecimento de painéis de notificações que informem o jogador de alguma mudança ou acontecimento que ocorra durante o jogo.

Os elementos que compõem a aplicação são apresentados no próximo sub-capítulo, juntamente com as cores que os caracterizam.

4.4. Cores da Aplicação

Para a escolha das cores da aplicação foi tida em conta a faixa etária do público-alvo, tendo sido decidido que estas deveriam ser contrastantes e chamativas de forma a cativarem o interesse da criança.

Foram escolhidas cores básicas, tais como o azul, amarelo, verde, laranja, castanho, entre outras. Estas cores conseguem contrastar entre si de forma harmoniosa e captam mais facilmente a atenção do jogador. A inspiração para a escolha das cores vem de uma referência visual [63].

De forma a não se tornar demasiado colorido e confuso, no ecrã de jogo foram diferenciadas as áreas de animação do personagem e de construção da solução através da utilização de cores neutras na segunda área. Sendo uma área de contrução, o fundo deve ser o mais simples possível de forma a não criar confusão e cansaço ao jogador quando este está a pensar e raciocinar sobre o problema.

De seguida são apresentados os diversos elementos que fazem parte da aplicação e descritas as cores que os compõem.

4.4.1. Personagens

Figura 31. Duarte

#468770	#000000
#55a38a	#35210f
#78c4a8	#f7c09a
#85d2ba	#efeded
#ffffff	

Figura 32. Blu

#4e3318
#c5770a
#df9319

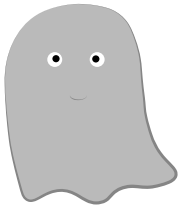


Figura 33. Fantasma

#000000
#848484
#bababa
#ffffff

4.4.2. Objectos

Figura 34. Planta

#628420
#79a828
#80be1f

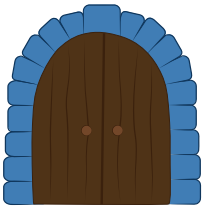


Figura 35. Porta

#0f365d
#3f7cb6
#3a200c
#482a10
#4e3318

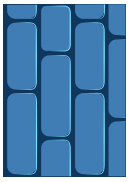


Figura 36. Parede

#0f365d
#3f7cb6
#7ae1ff

Figura 37. Cadeado e chave

#b75700
#ef8318
#ffffff

Figura 38. Martelo

#35210f
#777777
#8e8e8e
#c1c1c1

Figura 39. Chão

2d180a
#482a10
#4e3318

4.4.3. Botões

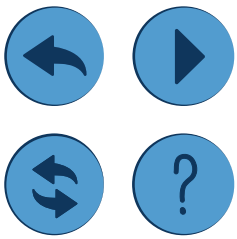
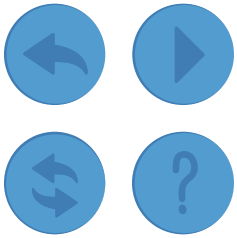


Figura 40. Botões activos

#0f365d
#529cce



#3f7cb6
#529cce

Figura 41. Botões inactivos

4.4.4. Área de construção

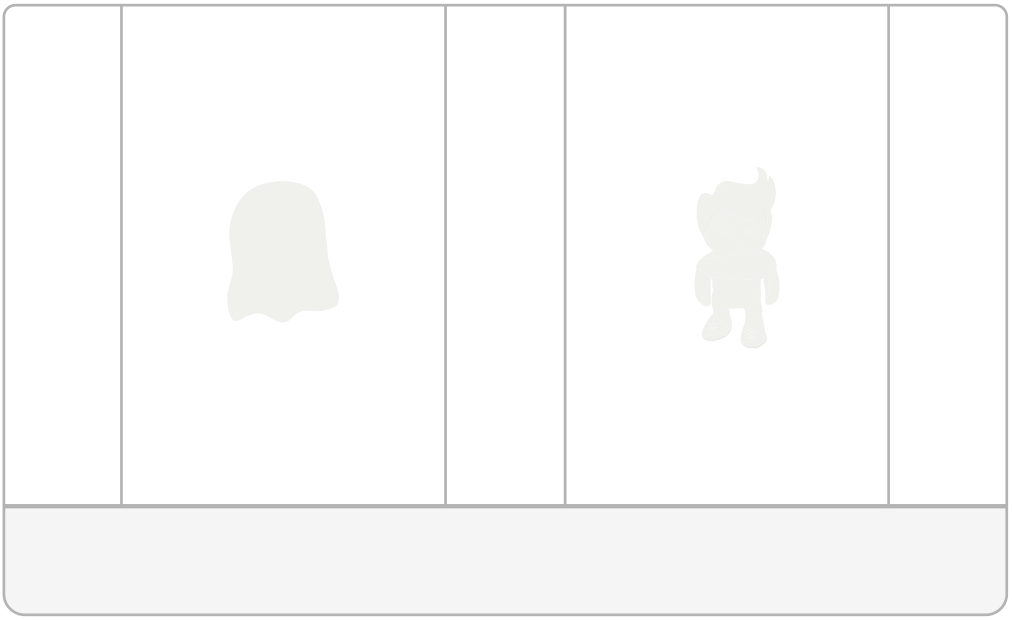


Figura 42. Área de construção das regras

#b3b3b3
#f5f5f5
#ffffff

4.4.5. Acções

Figura 43. Acções não seleccionadas

#0f365d
#f5f5f5

Figura 44. Acções seleccionadas

#0f365d
#df9319
#f5f5f5

4.4.6. Setas

Figura 45. Seta não destacada

#0f365d

Figura 46. Seta destacada

#df9319

4.4.7. Níveis

#3a200c
#55a38a
#c5770a

Figura 47. Nível fechado

#3a200c
#55a38a
#fffff

Figura 48. Nível aberto

#3a200c
#55a38a
#c5770a
#fffff

Figura 49. Nível concluído

4.4.8. Painéis de Notificação

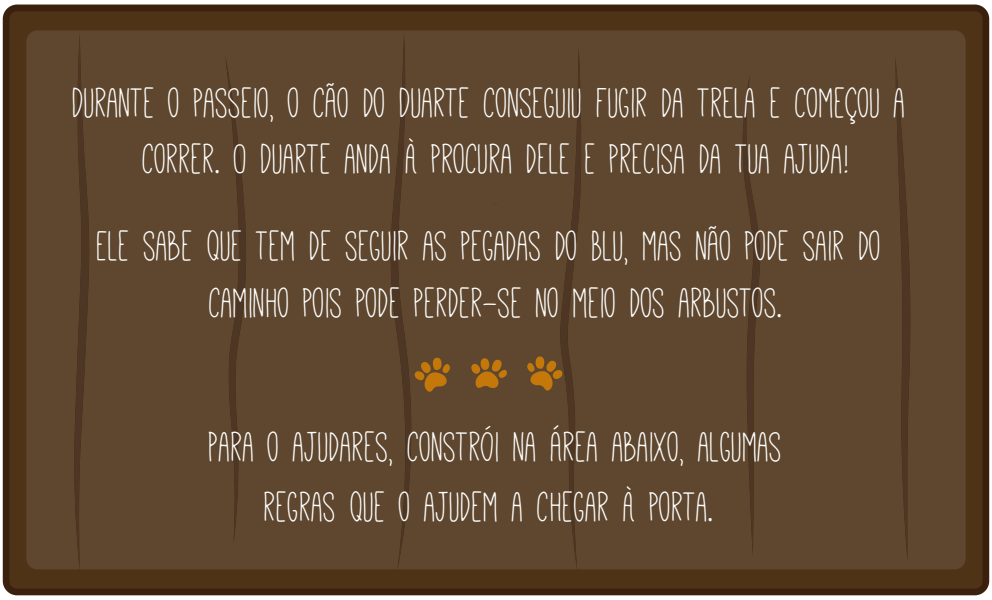


Figura 50. Painel de notificação

#3a200c
#55a38a
#c5770a
#ffffff

4.5. Ecrãs da Aplicação

Após terem sido realizados os testes ao protótipo em papel, foi necessário perceber se a visualização do ecrã de jogo no tablet se comportava do mesmo modo que em papel. Isto é, se os objectos que compõem o ecrã de jogo possuiam um tamanho razoável de modo a serem perceptíveis e facilmente seleccionados. A resolução de tablet definida para o desenvolvimento deste projecto foi a de 2048x1536 pixéis, sendo estas as dimensões do ecrã do tablet utilizado para testar a aplicação.

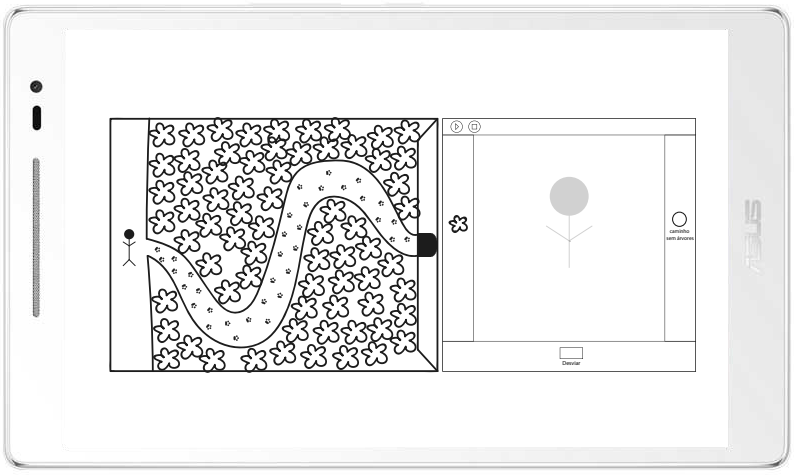


Figura 51. Protótipo de baixa fidelidade do ecrã de jogo no tablet

Observando o ecrã de jogo no ecrã do tablet, figura 51, foi possível constatar que a disposição deste teria de ser alterada. Ao serem dispostos numa posição horizontal, os objectos tornam-se mais pequenos do que é pretendido e perde-se algum espaço do ecrã que poderia ser melhor aproveitado como espaço para interagir com a definição da solução.

Foi, então, definida a orientação da aplicação como sendo vertical e iniciada a execução dos ecrãs de alta fidelidade, isto é, os ecrãs com o aspecto visual que a aplicação final teria.

4.5.1. Ecrã Inicial

O ecrã inicial, figura 52, é composto por vários elementos. No canto superior esquerdo encontra-se o nome da aplicação, tendo sido colocado de forma a ser facilmente identificável e legível. Abaixo deste, encontram-se vários elementos que fazem parte da história do jogo e que contam o seu início.

São apresentados o Duarte e o Blu; a entrada da casa abandonada onde o Blu entra após fugir do Duarte; o percurso que o Duarte percorre no primeiro nível até conseguir entrar na casa abandonada, representado pelas plantas; e o fantasma que faz parte de alguns níveis do jogo.

Por fim, são dispostos no centro do ecrã os três botões, *Jogar, Como Jogar* e *Sair*, definidos no sub-capítulo 4.2., que direccionam o jogador para um novo ecrã.

Figura 52. Ecrã Inicial

4.5.2. Ecrã de Níveis

O ecrã de níveis, figuras 53 e 54, apresenta os diversos níveis da aplicação. Inicialmente, o jogador só possui o nível 1 desbloqueado, sendo o único que é possível jogar. Os restantes são apresentados com um cadeado e vão sendo desbloqueados à medida que o jogador vai concluindo os níveis anteriores. Ao serem desbloqueados, o cadeado desaparece e aparece o número correspondente ao nível.

Caso o jogador pretenda voltar para o menu inicial, este pode fazê-lo clicando no botão que se encontra no canto superior esquerdo.

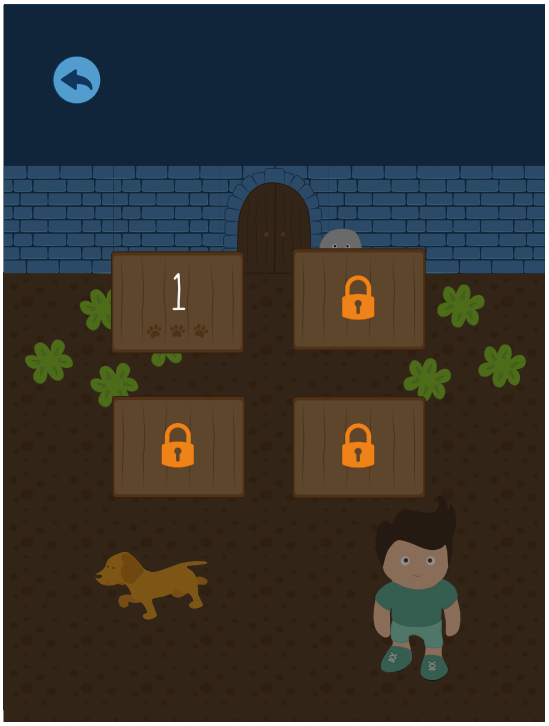


Figura 53. Ecrã de Níveis - nível 1 aberto

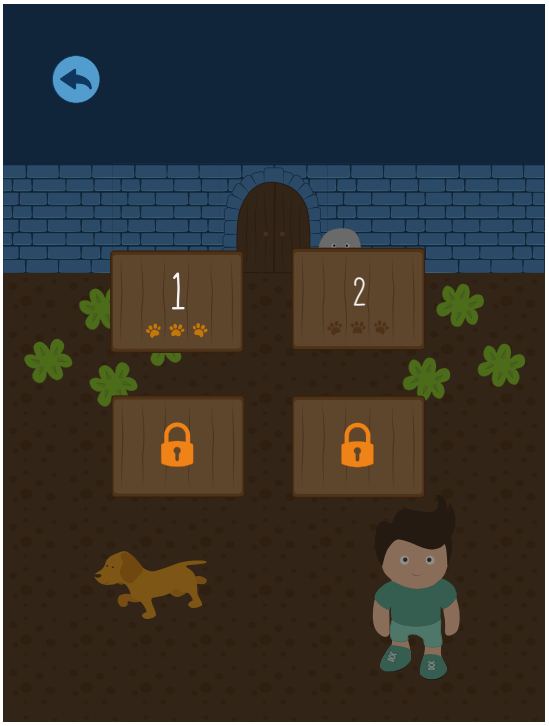


Figura 54. Ecrã de Níveis - nível 1 concluído e nível 2 aberto

4.5.3. Ecrã de Jogo

O ecrã de jogo, figura 55, sofreu alterações desde o protótipo realizado em papel, e foi desenhado para ocupar o ecrã do tablet na orientação vertical. Este ecrã é composto por duas áreas distintas, a área de jogo e a área de construção das regras, e por três ou quatro botões, dependendo dos níveis.

O botão que se encontra no canto superior esquerdo permite ao jogador retornar ao ecrã de níveis. À sua direita, encontram-se outros dois botões. O primeiro, permite ao jogador visualizar o resultado das regras construídas, e o segundo permite ao jogador alterar as regras criadas. Em alguns níveis, como o da figura 56, aparece um quarto botão que permite ao jogador receber ajuda sobre a construção da solução.

Abaixo dos botões são apresentadas as áreas de jogo e de construção. Na área de jogo, o jogador observa o resultado da solução, que realizou na área de construção, através da animação do personagem.

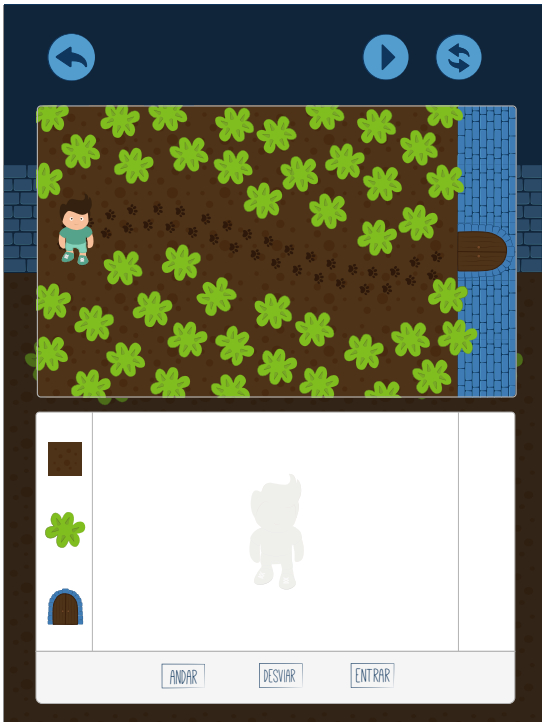


Figura 55. Ecrã de Jogo - nível 1

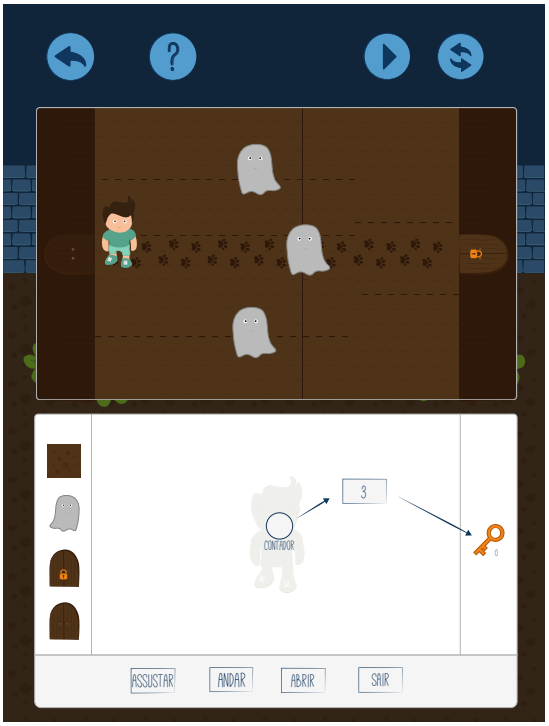
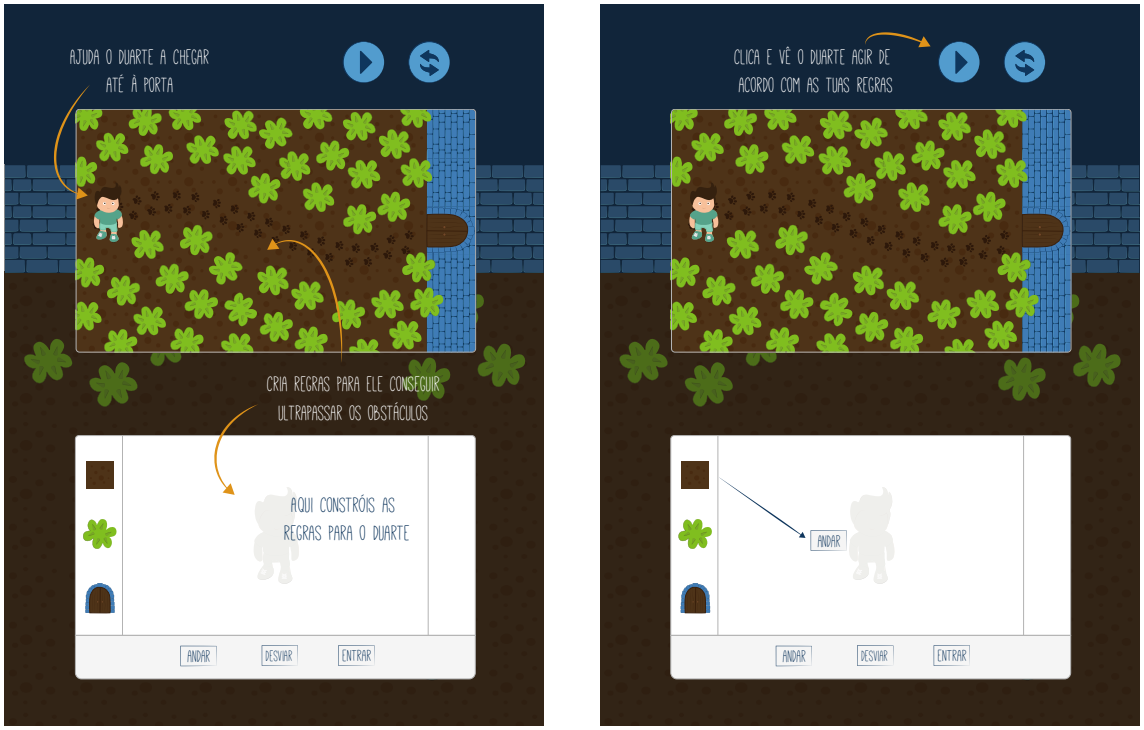


Figura 56. Ecrã de Jogo - nível 3

4.5.4. Ecrã de Como Jogar

Neste ecrã, são apresentados ao jogador os diferentes elementos que compõem o ecrã de jogo, e explicado como deve proceder para conseguir construir as regras e visualizar o seu resultado. Esta explicação é feita através de frases curtas e simples, de forma a não confundir o jogador.



Figuras 57 e 58. Ecrã de Como Jogar

5 - Implementação

Este capítulo foca-se no processo de implementação do protótipo funcional, desde as escolha da tecnologia até à descrição das funcionalidades implementadas.

No início deste capítulo é apresentada a estrutura da aplicação, sendo definida a liguagem de programação utilizada para o desenvolvimento deste projecto e os ficheiros que compõem a aplicação.

No segundo sub-capítulo é explicado o modo de construção das regras, assim como as regras que foram implementadas para que o jogo estivesse de acordo com as regras utilizadas na construção de Petri nets.

No terceiro e quarto sub-capítulos são introduzidos os *feedbacks* visuais e sonoros, explicando de que modo são realizados, quais as acções associadas a estes *feedbacks* e de que forma podem ajudar o jogador durante o jogo.

No quinto sub-capítulo são apresentados os tipos de notificações que foram implementadas no jogo e explicado qual o objectivo das mesmas.

No sexto sub-capítulo, referente a outras funcionalidades, é apresentado o método de recolha de informação implementado. Este método permite adquirir informação sobre as acções executadas pelo jogador durante a sua interacção com a aplicação.

5.1. Estrutura da Aplicação

O protótipo funcional foi implementado em linguagem Lua, e desenvolvido utilizando a plataforma Corona SDK. Esta plataforma permite o desenvolvimento de aplicações para diversos sistemas operativos, incluindo Android e iOS, que são os mais utilizados em tablets.

Foi criado um ficheiro .lua para cada um dos ecrãs definidos anteriormente, excepto para o ecrã de jogo. Este ecrã vai sofrendo alterações à medida que o jogador vai completando os níveis, tornando-se cada vez mais complexo e possuindo informações novas de nível para nível. Por estes motivos, foi definido um ficheiro .lua para cada nível, de forma a que cada ficheiro contenha o que é necessário ao nível que lhe corresponde.

Para que seja possível desbloquear um novo nível aquando da conclusão do nível anterior, foi necessário criar o ficheiro *mydata.lua* que serve como base de dados, guardando a informação sobre os níveis que o jogador vai completando. Por exemplo, ao ser concluído o nível 1, *level1* envia a informação de que foi concluído para *mydata*, que a guarda e envia, por sua vez, para o *levelMenu* de forma a este desbloquear o próximo nível.

Este modo de guardar a informação é temporário, uma vez que a informação guardada é elimidada quando se fecha a aplicação. O modo de guardar a informação foi definido desta forma para que seja mais simples reiniciar o jogo aquando da realização dos testes de usabilidade ao protótipo funcional.

A representação da estrutura do código da aplicação pode ser observada através da figura 59.

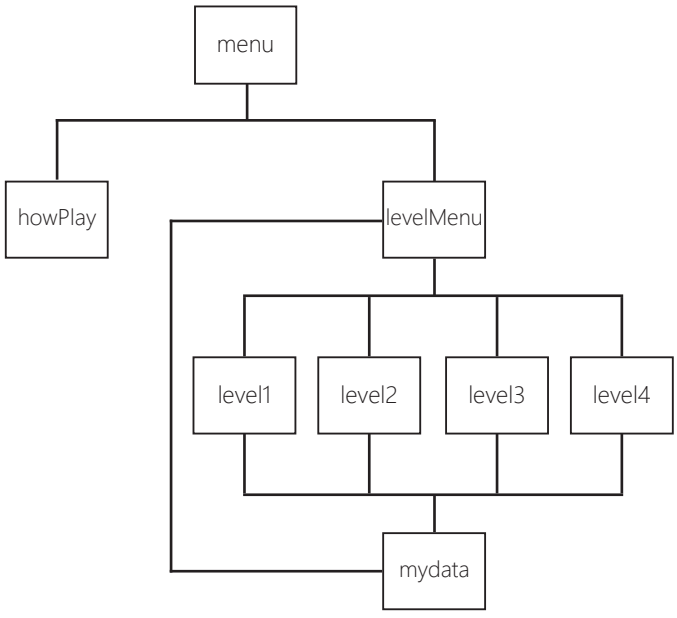


Figura 59. Estrutura do código da aplicação

5.2. Construção das Regras

A solução do problema colocado ao jogador é criada na área de construção. Nesta área, o jogador pode executar várias acções, tais como: seleccionar acções e objectos que se encontram na área de construção, arrastar as acções para o centro da área, e ligar os objectos às acções de forma a construir regras.

Ao seleccionar um objecto com o dedo, e arrastando-o até à acção pretendida, o jogador consegue criar uma ligação entre esses dois items. No entanto, só o conseguirá fazer caso tenha, anteriormente, arrastado a acção para o centro da área de construção. Para isso, foram definidos limites em x e y para a movimentação das acções, de forma a que estas não pudessem ser arrastadas para fora da área de construção.

Ao serem movidas para o centro da área, as acções ficam fixas no local onde o jogador as colocou. Desta forma, o jogador consegue seleccioná-las e ligá-las a objectos para a criação de uma regra. Se não houvesse essa restrição, o jogador, ao arrastar o dedo até ao objecto pretendido, não iria conseguir ligar a acção ao objecto, mas sim andar com a acção por toda a área. Ao ser criada uma regra, é colocada uma seta entre o objecto e acção seleccionada de forma a representar uma ligação entre os dois items.

Tendo em conta as regras para a construção de Petri Nets, foi implementada a regra de que os objectos apenas podem ser ligados a acções (figura 60), e as acções apenas se podem ligar a objectos (figura 61). Caso o jogador tente ligar duas acções ou dois objectos, não aparecerá uma seta a ligá-los. Desta forma o jogador percebe que esta acção não é possível de executar.

Figura 60. Ligação de um objecto a uma acção

Figura 61. Ligação de uma acção a um objecto

5.3. Feedback Visual

Uma vez que se pretende facilitar a aprendizagem através da utilização de *feedback* visual, este foi implementado de duas formas distintas. A primeira é feita através da animação dos objectos existentes na área de jogo, e a segunda é feita através do destaque das regras durante a animação.

A animação na área de jogo é feita de acordo com as regras criadas pelo jogador. Cada regra acciona uma acção, e esta acção pode ou não ser executada de acordo com o local onde se encontra o personagem. Por exemplo, se for criada uma regra em que o personagem deve saltar quando encontrar um buraco, mas o personagem não se encontra perto do buraco nesse momento, ele não vai executar essa acção.

Enquanto a animação decorre, são destacadas as regras que dão origem a essa animação na área de construção (figura 62). Este destaque é feito através da mudança de cor das setas e do contorno das acções e objectos, sendo alterado de azul para cor-de-laranja. O cor-de-laranja foi a cor escolhida devido a ser uma cor forte e com grande destaque perto do azul, facilitando a sua visualização por parte do jogador.

Não tão direccionado para a aprendizagem, mas também importante como *feedback* para a percepção do funcionamento do ecrã de jogo, é a alteração das cores dos botões, objectos e acções quando são seleccionados.

Ao ser seleccionado um botão, este altera a sua cor de forma a que o jogador se aperceba que este se tornou inactivo. Volta depois à sua cor inicial quando retorna ao seu estado activo (figura 63). Por sua vez, os objectos e acções alteram a cor do contorno para cor-de-laranja quando são seleccionados pelo jogador. Uma vez que o jogo é para tablet, este *feedback* é importante para que o jogador perceba se consegue seleccionar o que pretende ou não.

Figura 62. Regra destacada

Figura 63. Botões: inactivo e activo

5.4. Feedback Sonoro

O *feedback* visual é importante, mas pode ser complementado com *feedback* sonoro de forma a facilitar a percepção do jogador para o que vai acontecendo no ecrã. Atribuindo um som a um determinado acontecimento, o jogador toma conhecimento de que algo aconteceu mesmo que não se tenha apercebido imediatamente disso visualmente.

Ao todo foram utilizados dez ficheiros de áudio, obtidos em *www.freesound.org*. Estes sons foram atribuídos a acções que o jogador pode executar no jogo, a acções que o personagem executa na área de jogo, ao destaque das regras criadas na área de contrução e ao sucesso na resolução do desafio.

Acções do jogador

- Seleccionar um botão do jogo: som de um clique para que o jogador se aperceba que seleccionou o botão que pretendia;
- Seleccionar um objecto: som diferente do anterior, mas com o mesmo propósito;
- Arrastar uma acção: o som é reproduzido ao seleccionar e ao largar a acção, de forma a dar uma sensação de conclusão da acção no final;
- Criar uma regra: som semelhante a um clique, mas diferente dos anteriores. É reproduzido quando aparece a seta a ligar o objecto à acção e, tal como o som anterior, pretende dar uma sensação de conclusão da tarefa.

Acções do personagem

- Duarte a assustar um fantasma: "Boo!" é o som reproduzido quando esta animação é activada e pretende dar voz ao personagem quando este assusta o fantasma;
- Fantasma a atravessar a parede: o som reproduzido é semelhante ao som convencionado pela indústria do entretenimento à figura de um fantasma;
- Duarte a partir a parede: som de pedras a cair. Dá a entender que a parede foi partida e por isso desaparece um bocado dela.

Destaque das regras aquando da animação

Durante a animação do personagem na área de jogo, o jogador vai focar-se nessa área do ecrã e pode não se aperceber do destaque das regras aquando da animação. Para que ele se aperceba que algo está a acontecer na área de construção, é reproduzido o som referente à criação de regras por cada vez que uma regra é destacada.

Resolução do problema

Ao contruir uma solução para o problema, o jogador é levado a verificar se a solução criada é a correcta através da animação do personagem. Caso o personagem não alcance o objectivo definido, o jogador sabe que a sua solução não é a correcta. Esta conclusão é também transmitida através de um som que é reproduzido quando o personagem acaba a sua animação, dando a conhecer ao jogador que aquele é o final da animação e que deve criar uma nova solução. Por outro lado, quando o jogador é bem sucedido e a sua solução está correcta, é reproduzido um outro som que o leva a concluir, juntamente com o aparecimento de uma notificação, que conseguiu resolver o problema.

5.5. Notificações

As notificações vão aparecendo ao jogador ao longo dos níveis, facultando-lhe informação útil sobre o modo de interagir com o ecrã de jogo (figura 64); dando-lhe indicações para a construção das regras (figura 65); fazendo uma introdução a cada nível (figura 66); e informando o jogador de que completou o nível com sucesso (figura 67). Estas notificações têm como objectivo servir de apoio ao jogo para que o jogador não se sinta perdido em momento algum.

Figura 64. Notificação sobre o modo de interagir com o ecrã de jogo

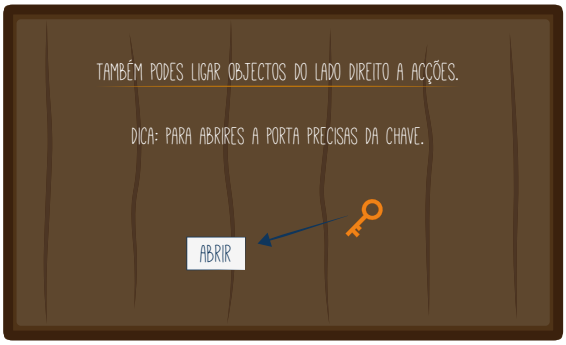


Figura 65. Notificação sobre o modo de construir as regras

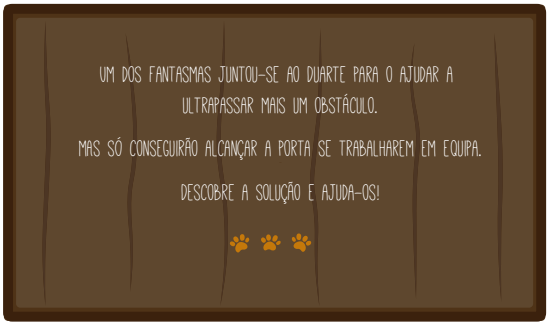


Figura 66. Notificação de introdução ao nível 4

Figura 67. Notificação de conclusão de nível

5.6. Outras Funcionalidades

A pensar nos testes de usabilidade e na forma como seria recolhida a informação sobre as acções tomadas pelo jogador durante o teste à aplicação, foi decidido criar um ficheiro .csv onde são guardadas todas as acções tomadas pelo jogador durante o jogo.

Os dados contidos neste ficheiro são separados por vírgula, e cada acção é descrita numa nova linha. Cada linha do ficheiro contém a seguinte informação:

- **número do jogador:** é atribuído um número ao jogador para que possam ser identificadas as acções tomadas por cada um;
- **ecrã:** é descrito qual o ecrã em que o jogador se encontrava quando realizou uma determinada acção;
- **objecto seleccionado:** é descrito qual o objecto que foi seleccionado pelo jogador no ecrã descrito anteriormente. Este objecto pode ser um botão, uma notificação, uma acção ou um objecto;
- **acção:** é descrita qual a acção realizada sobre o objecto. O jogador pode apenas seleccionar o objecto, retirando o dedo do tablet de seguida; ou pode arrastar o dedo após seleccionar o objecto;
- **objecto final:** é descrito qual o objecto que foi seleccionado no final da acção. Por exemplo, quando o jogador cria uma regra, ele vai ligar o **objecto seleccionado** ao **objecto final**. No caso de não se tratar da construção de uma regra, é colocado um **N** nesta variável com o significado de não existir um objecto final;
- **timestamp:** é descrito o tempo que passou desde a última acção tomadas pelo jogador, tornando possível visualizar quais as acções em que o jogador mais hesita. Também no final de cada nível é apresentado o tempo de execução desse nível.

Este ficheiro é criado e guardado numa directoria interna à aplicação, o que impossibilita o seu acesso através do tablet. Uma vez que se pretende obter este ficheiro para análise dos resultados obtidos, foi encontrada a solução de este ficheiro ser enviado pela aplicação através de um e-mail. Este e-mail é enviado quando o jogador selecciona no botão *Sair* do ecrã inicial, antes de ser fechada a aplicação.

6 - Avaliação

Após a implementação do protótipo funcional, foram realizados testes de usabilidade ao mesmo. Este capítulo aborda o método utilizado para a realização destes testes, assim como a análise realizada aos resultados obtidos.

Inicialmente são apresentados os perfis dos utilizadores que realizaram os testes, assim como os resultados que estes obtiveram durante a sua interacção com a aplicação. Posteriormente, é realizada uma análise desses resultados e retiradas conclusões acerca dos mesmo.

Por fim, são apresentadas as alterações realizadas ao protótipo funcional, necessárias para a resolução dos problemas descritos pelos utilizadores.

6.1. Testes de Usabilidade

Foram realizados testes de usabilidade ao protótipo funcional com nove utilizadores distintos, com idades compreendidas entre os 8 e os 26 anos. Os utilizadores possuiam diferentes graus de desenvolvimento da lógica, sendo na sua maioria utilizadores que possuem esta capacidade pouco desenvolvida.

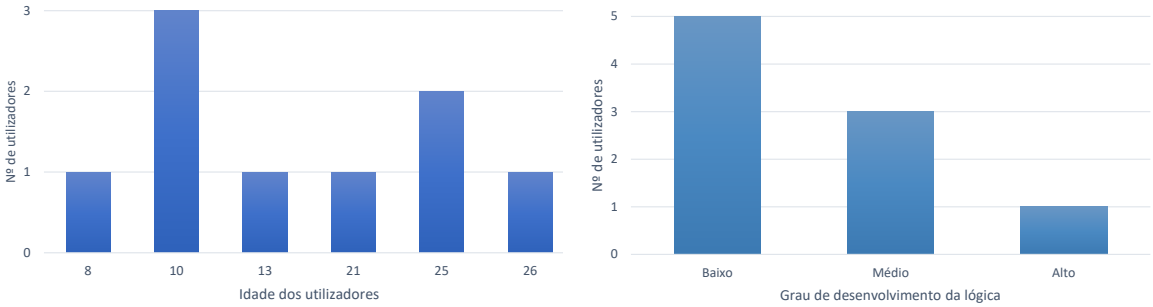


Figura 68. Idade dos utilizadores (esquerda) e grau de desenvolvimento da lógica dos utilizadores (direita)

O objectivo de realizar testes de usabilidade a crianças e a jovens adultos é perceber quais as diferenças que existem na aprendizagem da lógica em idades distintas, assim como perceber se esta aplicação poderia ser utilizada como ferramenta de aprendizagem para utilizadores com idades compreendidas entre os 20 e os 30 anos.

Para a realização dos testes foi disponibilizado um tablet com a aplicação instalada, dando ao utilizador liberdade para experimentar e interagir com a aplicação à sua vontade, sendo guardada toda a informação sobre as acções executadas pelo utilizador durante a interacção com a aplicação. Ao terminar de jogar, o utilizador respondia a algumas questões (anexo 1), avaliando a usabilidade da interface da aplicação, assim como o funcionamento do jogo.

Cada utilizador realizou o teste individualmente, de forma a não influenciar nas decisões dos outros jogadores durante a interacção com a aplicação (figura 69).

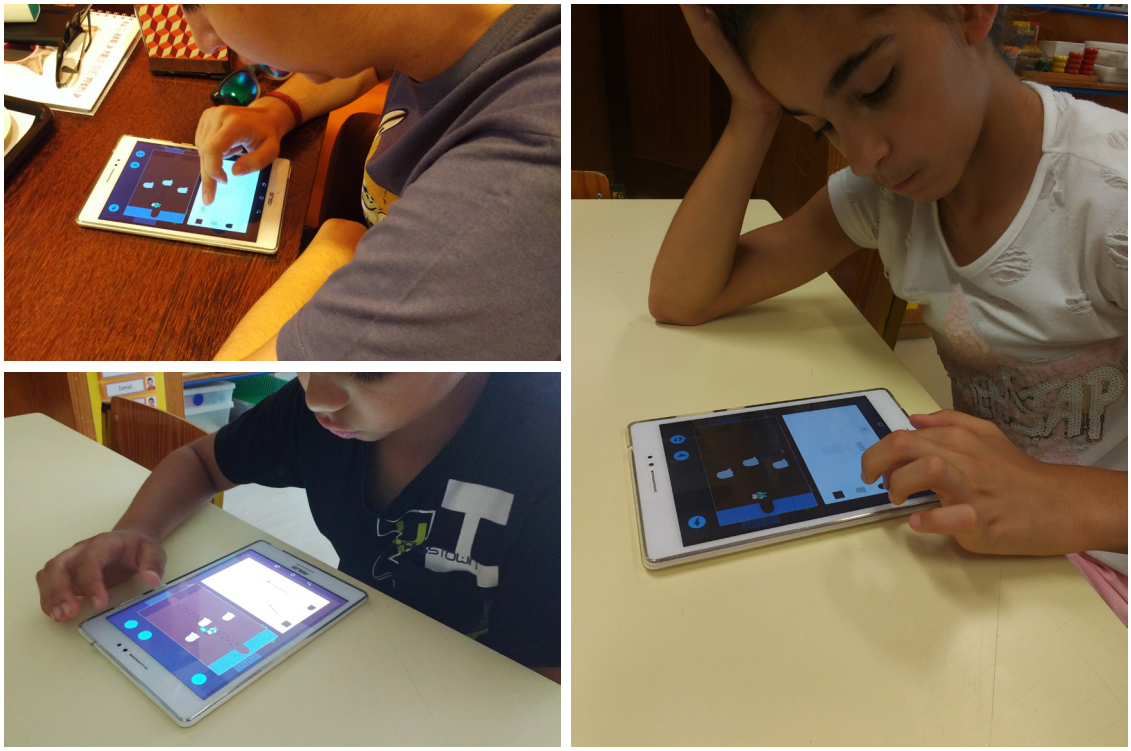


Figura 69. Utilizadores a interagir com o protótipo funcional

6.2. Análise de Resultados

Numa primeira fase, são analisadas as respostas dadas pelos utilizadores referentes à usabilidade da aplicação. Foram apresentadas diversas afirmações ao utilizador e este deveria dizer se concordava 'C' / não concordava nem discordava 'N' / não concordava 'NC' com a afirmação colocada. As afirmações são as seguintes:

1 - Os botões da aplicação são fáceis de identificar

2 - Percebe-se facilmente para que serve cada um dos botões

3 - No menu de níveis é fácil perceber quais os níveis que se podem jogar

4 - A introdução no início de cada nível ajuda a perceber o que é para fazer

5 - O nível 1 ajuda a perceber o modo de jogar

6 - Foi fácil perceber o que era preciso fazer para construir as regras

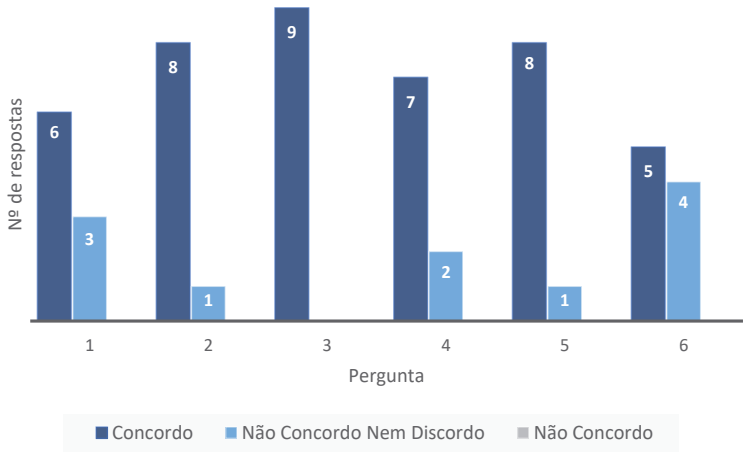


Figura 70. Respostas dos utilizadores às afirmações apresentadas

Após análise da figura 70, é possível concluir que os utilizadores não sentem dificuldade em interagir com a aplicação, identificam facilmente os botões existentes e percebem qual a função de cada um. Também o facto de sentirem que o nível 1 os ajuda a perceber o modo de jogo, facilitando a aprendizagem do modo de contruir as regras, permite-lhes direccionar o foco para a resolução do problema colocado.

Relativamente à história do jogo, todos os utilizadores sentiram vontade de ajudar o Duarte a encontrar o Blu, funcionando assim como motivação para resolver os desafios que lhes eram colocados.

Direccionando a análise para o funcionamento dos níveis que compõem a aplicação, foram analisadas as respostas dadas pelos utilizadores relativamente a cada um dos níveis.

No primeiro nível, todos os utilizadores sentiram que as animações e notificações apresentadas permitiam uma aprendizagem mais fácil do modo de jogar, bastando seguir as indicações para resolver este nível com sucesso. Os nove utilizadores completaram este nível com facilidade.

No segundo nível, uma vez que era semelhante ao primeiro, os utilizadores não sentiram dificuldade e completaram-no rapidamente. Neste nível, já foi possível terem contacto com a animação do Duarte e verificar se as regras construídas eram as correctas. Todos os utilizadores afirmaram que era fácil perceber quando tinham alguma regra incorrecta uma vez que a animação parava e a regra era destacada a cor-de-laranja.

No terceiro e quarto níveis, os utilizadores sentiram alguma dificuldade em construir as regras devido à existência de um novo elemento, o contador. A maioria dos utilizadores sentiu dificuldade em perceber para que servia o contador, e ao mesmo tempo não sabiam o que podiam ligar para criar uma regra. Mesmo utilizando o botão de ajuda existente nestes dois níveis, a informação que lhes era facultada não permitia perceber para que servia este elemento.

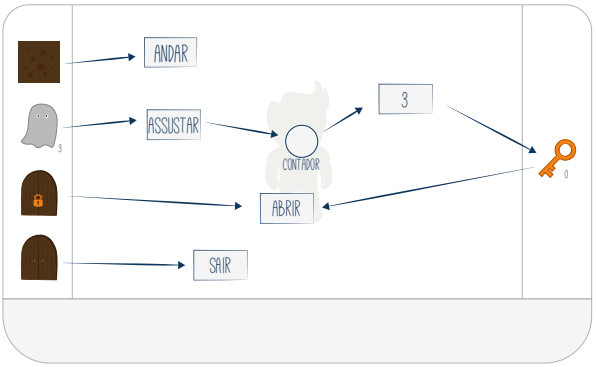


Figura 71. Solução do nível 3 utilizando o contador

A figura 72 apresenta a dificuldade que os utilizadores sentiram ao resolver cada um dos níveis, assim como o tempo médio de resolução de cada um. Este tempo médio de resolução é calculado utilizando os tempos descritos nos ficheiros .csv dos utilizadores.

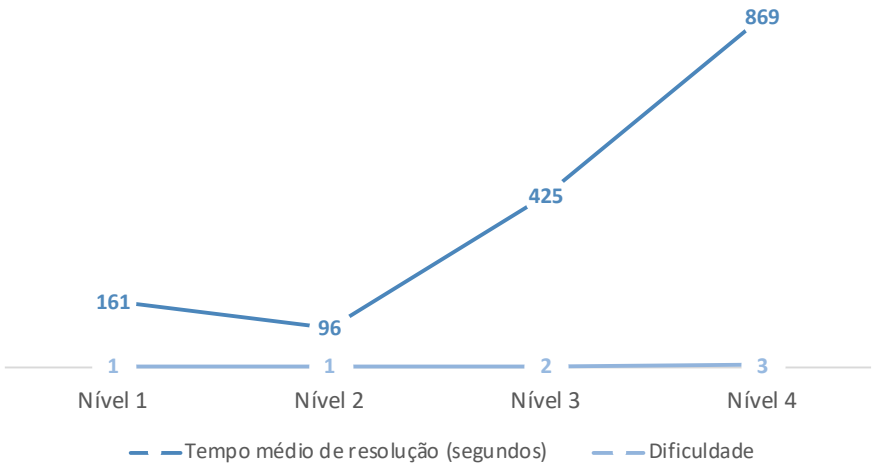


Figura 72. Tempo médio de resolução e dificuldade sentida em cada nível (1 - fácil, 2 - médio, 3 - difícil)

De modo a perceber quais as diferenças existentes entre as crianças e os jovens adultos relativamente à aprendizagem da lógica e à interacção com a aplicação, foram criados gráficos de médias ilustrativos da progressão acelerada da dificuldade. Foi calculado o número médio de acções realizadas em cada nível, assim como o tempo médio de resolução do mesmo, por cada grupo de utilizadores. O grupo das crianças é composto por cinco utilizadores, com idades compreendidas entre os 8 e os 13 anos, e o grupo dos jovens adultos é composto por 4 utilizadores, com idades compreendidas entre os 21 e os 26 anos.

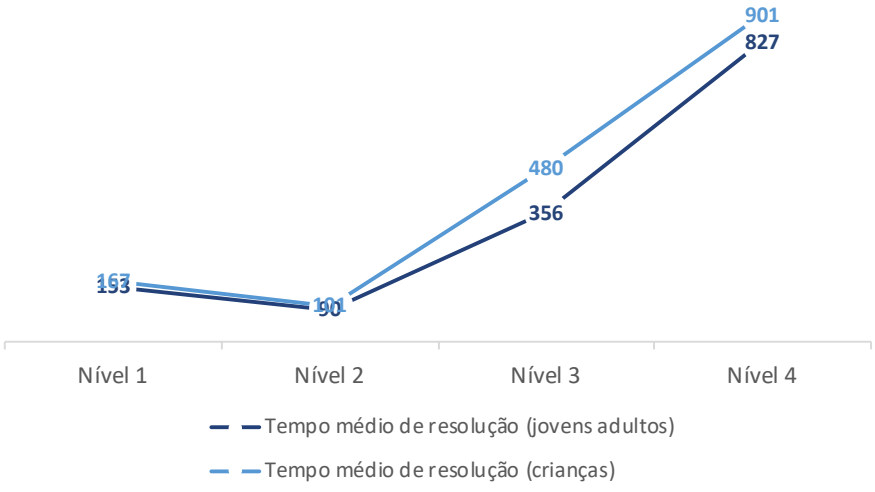


Figura 73. Comparação do tempo médio de resolução de cada nível

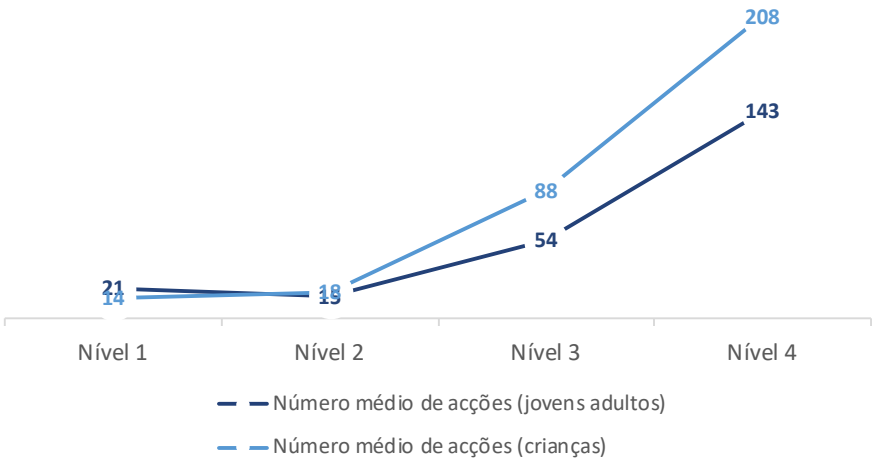


Figura 74. Comparação do número médio de acções realizadas em cada nível

Observando os valores apresentados nas figuras 73 e 74, é possível concluir que os utilizadores que possuem idades entre os 8 e os 13 anos tendem a demorar mais tempo a resolver o problema que lhes é colocado e, ao mesmo tempo, realizam mais acções do que os utilizadores com idades entre os 21 e os 26 anos.

Aliando os dados apresentados à observação feita durante os testes, conclui-se que os utilizadores mais jovens procuram encontrar a solução através de várias tentativas, tentando perceber o que acontece caso criem uma determinada regra. Desta forma, acabam por perceber quais as regras que devem criar através da observação da animação do Duarte e do destaque das regras.

Por sua vez, os utilizadores mais velhos tendem a perder mais tempo a tentar perceber quais as regras que podem e devem criar de modo a resolver o problema, o que faz com que realizem um menor número de acções do que os utilizadores mais novos.

No final do questionário, era dada a possibilidade ao utilizador de realizar observações acerca da aplicação. Uma observação feita pela maioria dos utilizadores foi o facto de, ao clicar no botão para alterar ou construir novas regras, a área de construção ficava a branco, tendo o utilizador de criar novamente todas as regras. Nos dois primeiros níveis, em que existem poucos objectos e acções, este problema não era sentido por parte dos utilizadores. Mas ao jogarem o terceiro e quarto níveis, com o aumento da complexidade, os utilizadores perdiam muito tempo a voltar a construir todas as regras e, caso se enganassem, teriam de voltar a construir tudo de novo.

Concluiu-se que este método de eliminar as regras criadas faz com que a motivação do utilizador diminua, ao mesmo tempo que permite que este se sinta confuso caso não se lembre de todas as regras que tinha criado antes de seleccionar o botão.

6.3. Alterações ao Protótipo Funcional

A partir dos resultados dos testes de usabilidade foi possível perceber quais as alterações que deveriam ser realizadas ao protótipo funcional de modo a melhorar a aprendizagem e a interacção com a aplicação.

A primeira alteração realizada ao protótipo funcional passou por modificar o modo de eliminar as regras criadas pelo jogador. O botão de alterar ou criar regras, anteriormente designado de Stop, passa a ter o nome de Undo e, ao clicar neste botão, o jogador verá eliminada apenas a última regra criada. Se clicar mais vezes no botão, vão sendo eliminadas as regras criadas mais recentemente.

Esta alteração foi realizada através da criação de uma lista onde vão sendo guardadas todas as acções que o jogador realiza na área de construção das regras. Quando o jogador selecciona o botão, é retirada a última acção da lista e eliminada a última regra criada (figuras 75 a 79).

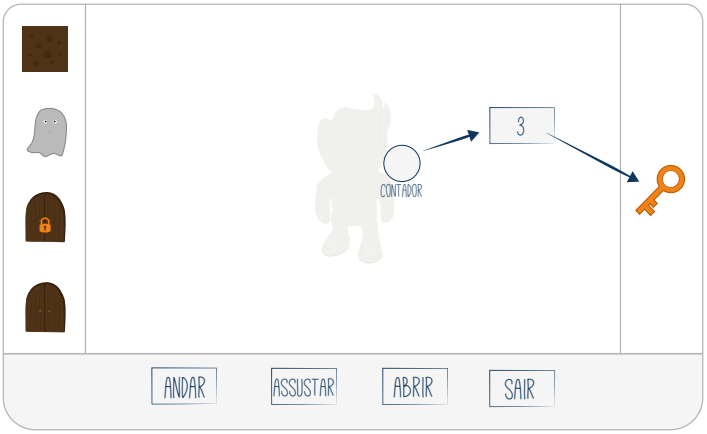


Figura 75. Área de construção sem regras criadas

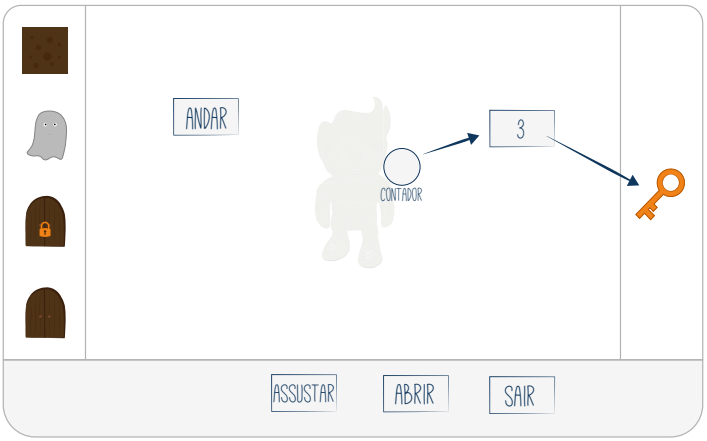


Figura 76. Área de construção após o jogador arrastar uma acção para o centro

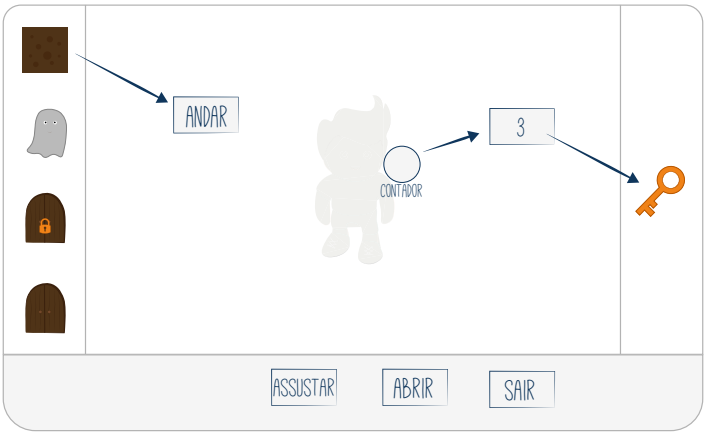


Figura 77. Área de construção após o jogador criar uma regra

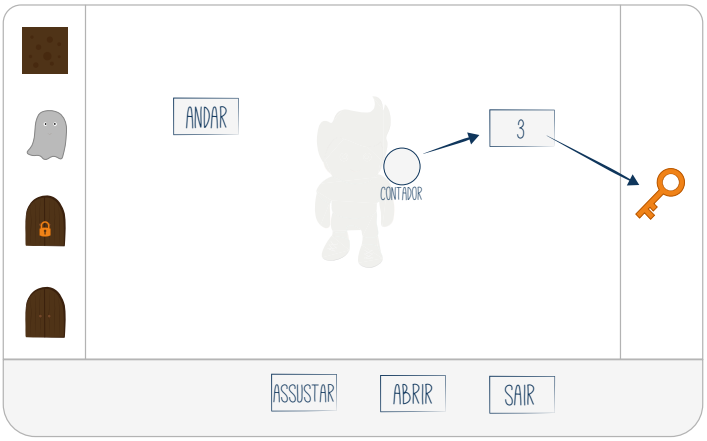


Figura 78. Área de construção após o jogador seleccionar o botão Undo.
A regra criada anteriormente é desfeita.

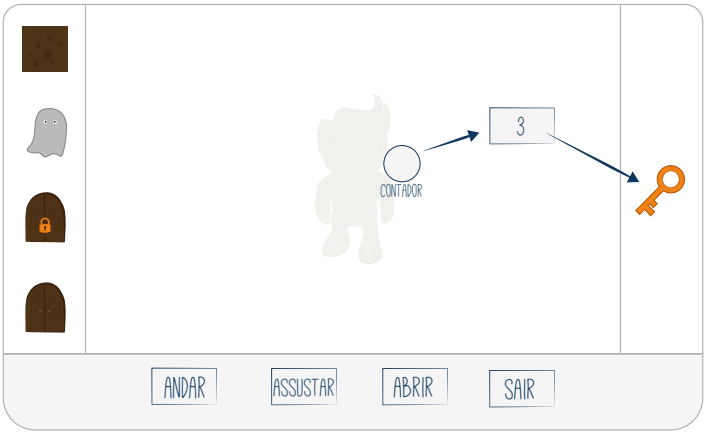


Figura 79. Área de construção após o jogador seleccionar novamente o botão Undo.
A acção 'andar' é colocada no local onde se encontrava inicialmente.

Outra alteração realizada ao protótipo foi a introdução de mais informação sobre o contador de forma ajudar o jogador a perceber como pode utilizar este elemento na construção das regras. Esta informação é fornecida ao jogador através de notificações que lhe dão pistas de como deve utilizar o contador nos níveis 3 e 4.

A figura 80 apresenta a notificação colocada nos níveis 3 e 4 que pretende ajudar o jogador a perceber o que pode ser ligado ao contador de forma a conseguir criar uma regra.

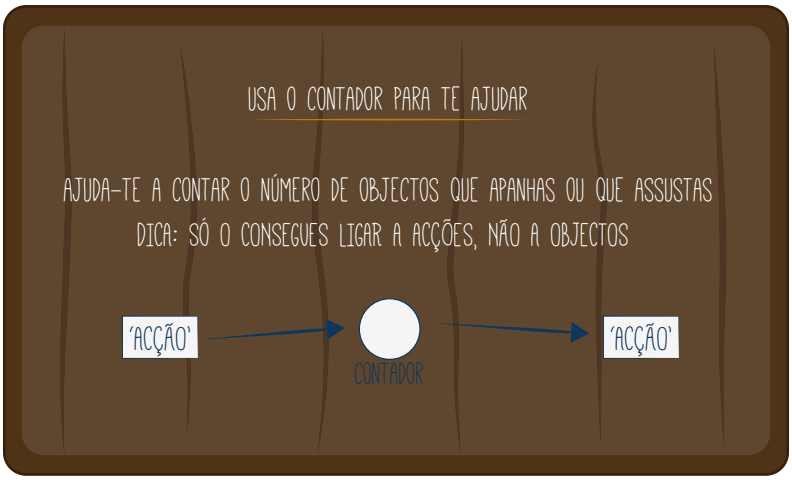


Figura 80. Notificação sobre o uso do contador

6.4. Trabalho Futuro

A aplicação Petri Dog foi criada para promover o desenvolvimento da lógica e da programação, e pretende-se colocá-la acessível ao público em geral. No entanto, mesmo estando a funcionar correctamente, o protótipo funcional deverá ser submetido a algumas melhorias de forma a tornar-se numa boa ferramenta de aprendizagem.

É necessário introduzir níveis intermédios, entre aqueles que já estão construídos, de forma a ir aumentando a dificuldade mais lentamente para que o jogador tome conhecimento e perceba, mais facilmente, para que serve cada elemento que é introduzido no jogo.

Também será necessário criar um sistema de base de dados que guarde a informação acerca dos níveis já resolvidos pelo jogador para que este não volte ao início do jogo cada vez que abre a aplicação.

Por fim, seria interessante criar novos níveis que introduzissem, progressivamente, o jogador a outros elementos da programação.

7 - Conclusão

A proposta de criação de uma ferramenta para a aprendizagem da lógica e da programação em crianças possibilitou o desenvolvimento de um artefacto que se diferencia das aplicações que já se encontram no mercado. Esta diferença deve-se ao facto de ser utilizado um método de construção da solução que se assemelha à construção de Petri Nets e que, devido à facilidade na leitura das regras, o jogador rapidamente aprende a construir as suas regras e direcciona o foco para a resolução do problema que lhe é colocado.

A aplicação Petri Dog fornece uma história simples, mas ao mesmo tempo motivadora para o jogador. Na procura pelo Blu, o jogador é levado a ajudar o Duarte a superar alguns obstáculos, motivando-o a continuar a jogar até conseguir encontrar o Blu. Sendo a motivação um factor importante para a aprendizagem, conclui-se que esta aplicação permite uma aprendizagem mais fácil, uma vez que consegue manter o jogador motivado ao longo dos níveis. Esta conclusão é baseada nos resultados observados durante os testes de usabilidade, onde os utilizadores mais jovens questionavam sobre a existência de mais níveis aquando do término do último nível do protótipo funcional.

Os níveis existentes na aplicação pretendem proporcionar ao jogador uma evolução progressiva da aprendizagem. No entanto, após os testes de usabilidade realizados, concluiu-se que seria positivo introduzir novos níveis entre o segundo e o terceiro níveis, e entre o terceiro e quarto níveis, de forma a permitir ao jogador consolidar melhor a informação sobre os novos elementos que vão sendo introduzidos ao longo do jogo. Ao perceber como funciona cada elemento existente no jogo, o jogador consegue manter a motivação e a vontade em continuar a jogar.

Por fim, observando os resultados obtidos nos testes realizados ao protótipo funcional por utilizadores de diferentes idades, conclui-se que esta aplicação pode ser utilizada, não só por crianças, mas também por adultos que queiram desenvolver a lógica.

Referências

[1] http://el.media.mit.edu/logo-foundation/what_is_logo/history.html

[2] http://www.nied.unicamp.br/oea/mat/LOGO_IMPLICACOES_bette_nied.pdf

[3] http://movetheturtle.com/

[4] http://web.media.mit.edu/~jmaloney/papers/ScratchLangAndEnvironment.pdf

[5] https://scratch.mit.edu/

[6] https://minecraft.net/pt/

[7] https://phaser.io/news/2015/12/minecraft-meets-hour-of-code

[8] https://studio.code.org/s/mc/stage/1/puzzle/1

[9] https://carlispina.wordpress.com/2013/06/24/daisy-the-dinosaur/

[10] https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/kodu-kodugrammar.pdf

[11] http://www.educationscotland.gov.uk/learningandteaching/approaches/ictineducation/gamesbasedlearning/gamedesign/kodu.asp

[12] http://www.slate.com/articles/technology/technology/2009/07/logo_on_steroids.html

[13] https://www.kodugamelab.com/

[14] https://www.sitepoint.com/learn-basic-programming-concepts-while-you-play-with-cargo-bot-for-ios/

[15] http://roboticsblog.org/2009/fun/robozzle-a-robot-programming-game/

[16] https://computinged.wordpress.com/2014/09/16/gidget-is-now-released-a-debugging-puzzle-game-for-novice-programmers/

[17] https://ischool.uw.edu/feature-stories/help-gidget-save-animals-and-clean-chemical-spill

[18] http://www.piaget.org/aboutPiaget.html

[19] Cook, J. L., Cook G., Child Development – Principles & Perspectives, Pearson College Division, 2009.

[20] http://www.mat.uc.pt/~guy/psiedu2/piaget

[21] http://www.helioteixeira.org/ciencias-da-aprendizagem/teoria-do-desenvolvimento-cognitivo-de-jean-piaget/

[22] https://en.wikipedia.org/wiki/Problem-based_learning

[23] H. S. Barrows, R. M. Tamblyn; Problem-Based Learning: An Approach to Medical Education, Springer Publishing Company, 1980.

[24] http://web.stanford.edu/dept/CTL/cgi-bin/docs/newsletter/problem_based_learning.pdf

[25] Klabbers, J. H. G., The Magic Circle: Principles of Gaming & Simulation. Sense Publishers, 2006.

[26] Huizinga, J., Homo Ludens: A Study of the play element in culture. Boston: The Beacon Press, 1955.

[27] Gee, J.P., Good Games Good Learning. Peter Lang Publishing Inc, 2007.

[28] Smed, J.; Hakonen, H., Towards a Definition of a Computer Game. Turku Centre for Computer Science, 2003.

[29] Adams, E., Fundamentals of Game Design, Third Edition. Pearson Education, 2014.

[30] Ulicsak, M., & Wright, M. (2010). Games in Education: Serious Games.Futuerlab Series.

[31] Vahldick, A.; Mendes, A.J.; Marcelino, M. J.; Hogenn, M.; Schoeffel, P. Testando a Diversão em um Jogo Sério para o Aprendizado Introdutório de Programação, DEI, Universidade de Coimbra.

[32] Susi, T., Jahaneeson, M., & Baclund, M. (2007). Serious Games – An Overview. Skövde, Sweden: School of Humanities and Informatics, University of Skövde.

[33] Lopes, N., Oliveira, I. (2013). Videojogos, Serious Games e Simuladores na Educação: usar, criar e modificar. Universidade Aberta

[34] http://www.scholarpedia.org/article/Petri_net
Reising, W., Grzegorz,R., Lectures on Petri Nets I: Basic Models – Advances in Petri Nets. Spring-Verlag Berlin Heidelberg, 1998. pp 1-11.

[35] Araújo, M., Roque, L., Modeling Games with Petri Nets. 2009.

[36] https://www.informatik.uni-hamburg.de/TGI/PetriNets/introductions/aalst/

[37] Pereira, L., Roque, L., Understanding the Videogame Medium through Perspecti ves of Participation. 2013.

[38] Pereira, L. L., Roque, L. G.; Design Guidelines for Learning Games: the Living Forest Game Design Case. 2009.

[39] http://datapeak.net/images/logoscn1.jpg

[40] http://www.annehelmond.nl/wordpress/wp-content/uploads/2007/11/logo_mit.png

[41] http://a5.mzstatic.com/us/r30/Purple1/v4/f2/a4/ba/f2a4bad8-8113-960f-8bfe-df6dcbab9601/screen480x480.jpeg

[42] http://news.mit.edu/sites/mit.edu.newsoffice/files/images/2013/20130514110054-1_0_0.jpg

[43] https://llk.media.mit.edu/files/content/projects/scratch-editor.png

[44] http://geeksroom.com/wp-content/uploads/2015/11/minecraft-hour-of-code.jpg

[45] http://1.bp.blogspot.com/-27Bq_zvjQWg/VlW9pgg1ScI/AAAAAAAAVys/gbu4agJ8LLo/s1600/Screen%2BShot%2B2015-11-25%2Bat%2B8.44.48%2BAM.png

[46] http://www.educationalappstore.com/images/screenshots/app10020/4.jpeg

[47] http://classtechtips.com/wp-content/uploads/2013/07/image-68.png?w=300

[48] https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/kodu-programming_ui.jpg

[49] www.globalnerdy.comwordpresswp-contentuploads20100211moveselected.jpg

[50] http://sites.ssis-suzhou.net/hourofcode/files/2015/12/Photo-Jul-16-2-05-26-PM.png

[51] https://lh3.googleusercontent.com/_zz7PQdRGySXEHSgRwvHTQ_nJMpZbeOKftzX5A72XHk8KLgaHrWDHkqYpDB027BYPQ=h900

[52] http://1u88jj3r4db2x4txp44yqfj1.wpengine.netdna-cdn.com/wp-content/uploads/2014/05/robozzle-screen.png

[53] http://allusefulinfo.com/content/wp-content/uploads/2015/02/Robozzle.jpg

[54] http://www.helpgidget.org/media/tutorial/tutorial_introduction-01.png

[55] http://www.nndb.com/people/359/000094077/

[56] http://www.unicamp.br/iel/site/alunos/publicacoes/textos/d00005.htm

[57] https://www.simplypsychology.org/bruner.html

[58] http://www.psicologiaexplica.com.br/jerome-bruner/

[59] http://www.eses.pt/usr/ramiro/docs/etica_pedagogia/A%20Pedagogia%20de%20JeromeBruner.pdf

[60] www.papert.org

[61] https://www.researchgate.net/figure/259146248_fig3_Figure-3-Design-science-research-methodology-process-model-Peffers-et-al-2007

[62] https://sites.google.com/site/onlinegeocaching/key-ideas-of-scaffolding-in-the-design

[63] https://img.itch.zone/aW1hZ2UvMjQyNi81Njk3MS5wbmc=/original/eCcFAZ.png

Anexo 1

Petri Dog Teste de Usabilidade

Idade: _____

Responde às perguntas colocando um X na coluna que corresponde à tua opinião.

Afirmação	**NC**	**N**	**C**
Os botões da aplicação são fáceis de identificar			
Percebe-se facilmente para que serve cada um dos botões			
No menu de níveis é fácil perceber quais os níveis que se podem jogar			
A introdução no início de cada nível ajuda a perceber o que é para fazer			
O nível 1 ajuda a perceber o modo de jogar			
Foi fácil perceber o que era preciso fazer para construir as regras			
Completei o primeiro nível com facilidade			
Completei o segundo nível com facilidade			
Completei o terceiro nível com facilidade			
Completei o quarto nível com facilidade			
Senti vontade de continuar a jogar depois de terminar os níveis			

NC – Não Concordo | N – Não Concordo Nem Discordo | C - Concordo

Responde às questões de acordo com o que sentiste ao jogar o Petri Dog.

1- Percebeste a história do jogo? De que forma é que a história te motivou para continuares a jogar?

_____
_____
_____

2- De que forma o nível 1 te ajudou a aprender a jogar este jogo?

_____
_____
_____

3- Relativamente à construção das regras, com que facilidade conseguiste perceber como se construíam e o que significava cada uma?

_____
_____
_____

4- Depois de construíres as tuas regras, o Duarte movimenta-se de acordo com essas regras. Como é que a animação do Duarte te ajudou a perceber o que fazia cada uma das regras que tinhas construído?

_____
_____
_____

5- Nos níveis 3 e 4 é introduzido um novo elemento (contador). Qual a facilidade que tiveste em perceber qual o seu propósito e como funcionava?

_____
_____
_____

6- Existe, nos níveis 3 e 4, um botão de ajuda. Clicaste nele? Se sim, de que forma é que ele te ajudou a construir as regras?

_____
_____
_____

Outras observações:

_____
_____
_____
_____
_____