UNIVERSIDADE D
COIMBRA

Pedro Duarte de Almeida Estanqueiro e Cunha de Carvalho

VISUALIZAÇÃO INTERATIVA DE PROGRAMAS ESCRITOS EM PROCESSING

VOLUME 1

Relatório final no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software, orientada pelos Professores António José Mendes e Pedro José Mendes Martins e apresentada à Faculdade de Ciências e Tecnologia/Departamento de Engenharia Informática.

Outubro de 2021

Esta página foi propositadamente deixada em branco.

Faculdade de Ciências e Tecnologias
Departamento de Engenharia Informática

Visualização Interativa de Programas Escritos em Processing

Pedro Duarte de Almeida Estanqueiro e Cunha de Carvalho

Relatório final no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software, orientada pelos Professores António José Mendes e Pedro José Mendes Martins e apresentada à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática.

Outubro de 2021

UNIVERSIDADE D
COIMBRA

Esta página foi propositadamente deixada em branco.

Acknowledgements

To my supervisors, António José Mendes and Pedro José Mendes Martins, for the availability, guidance, understanding and help throughout this process, you were essential in the success of this project.

To Mrs. Anabela Reis, librarian of building 2, to which I promiss this acknowledgement in exchange for the borrowing of a book for a year.

To my dad, for never letting me go down in the hardest times, to my mom for all the help and all the time spent on me, to both for all the support in all my life decisions, as well as to my three sisters which I proudly say that I am their brother. And to my dear grandparents.

Finally, to all the people that were apart of this journey, in Tomar, Coimbra, inside or outside the DEI, in Portugal, or outside of Portugal and especially two people: Francisco and Cátia, thank you for never leaving my side.

Esta página foi propositadamente deixada em branco.

Agradecimentos

Aos meus orientadores, António José Mendes e Pedro José Mendes Martins, pela disponibilidade, pelo acompanhamento, pela compreensão e ajuda durante todo o processo, foram essenciais para o sucesso deste projeto.

À senhora Anabela Reis, funcionaria da biblioteca do pólo 2 a quem prometi fazer este agradecimento pelo empréstimo de um livro, durante um ano.

Ao meu pai por nunca me ter deixado ir abaixo nos momentos mais difíceis, à minha mãe por toda ajuda e todo o tempo que sempre disponibiliza para mim e a ambos por todo o apoio em qualquer decisão da minha vida, assim como as minhas três irmãs de quem tenho muito orgulho de ser irmão. E aos meus queridos avós.

Por fim, a todos aqueles de fizeram parte da minha vida nesta caminhada, em Tomar, em Coimbra, dentro e fora do DEI, em Portugal, fora de Portugal e em especial a duas pessoas: Francisco e Cátia, obrigado por nunca saírem do meu lado.

Esta página foi propositadamente deixada em branco.

Abstract

Processing is a programming language increasingly used for the introduction of programming languages, mainly utilised in fields such as Visual Arts and related to IT, therefore progressively more required. Throughout this dissertation there was a thorough investigation, research and development towards enforcing this concept. In order to achieve this basis, we proceeded to the design of a translator, which integrated with a Tool of IDE Processing, originated an interactive visualisation of written programs in Processing, achieving the end goal.

The use of tech tools for the support of programme learning/teaching, combined with traditional learning/teaching, has been subject of many difficult questions and obstacles. However, throughout the 21st century there was an evident rise on the employment of these types of tools. Nevertheless, more interest, studies and analysis, emerged in this matter which is as complex as it is interesting simultaneously.

The following report assembles a description and analysis of the developed work made through this internship, which forced the familiarisation with processing language and its IDE, in order to establish the basis of knowledge in Processing, to identify prime problems found in the teaching of programming languages, as well as difficulties and hardships that students and peers face and in addition, the reason why they face them. Furthermore, this paper accounts for the information about the final product, the IDE processing tool and its detailed procedure, reaching the final aim.

In addition, this dissertation approaches the trial tests made on the final product culminating results and conclusions. Finally, a general and final description is made on the work developed throughout these fourteen months with positive prospects on future work.

Keywords

Processing, Translator, Compiler, IDE Processing, Tech Tools, Introduction to Programming, Interactive Visualisation, Teaching, Students' Misconceptions and Difficulties, Programming Assistance, Learning.

Esta página foi propositadamente deixada em branco.

Resumo

Processing é uma linguagem de programação, cada vez mais utilizada para a introdução de linguagens de programação, sendo utilizada sobretudo em áreas como artes visuais e relacionadas com informática e por isso, mais requisitada. No decurso desta dissertação, estudou-se, investigou-se e desenvolveu-se o suporte à sua aprendizagem. Para fazer este suporte procedeu-se à criação de um tradutor, que integrado com uma ferramenta do IDE Processing deu origem à visualização interativa de programas escritos em Processing que se pretendia.

O uso de ferramentas tecnológicas para o auxílio do ensino à programação, em conjunto com o ensino tradicional, tem vindo a ser um tema que coloca muitas perguntas e dificuldades, porém a partir do século vinte e um notou-se uma crescente utilização deste género de ferramentas. Como tal, mais interessados, estudos e análises têm surgido sobre este assunto, tão complexo e interessante ao mesmo tempo.

O presente relatório reúne uma descrição e análise sobre o trabalho desenvolvido ao longo de todo este estágio, que obrigou à familiarização com a linguagem Processing e o seu IDE, para que fosse possível estabelecer as bases de conhecimento em Processing, identificar os principais problemas encontrados no ensino das linguagens de programação, assim como as dificuldades e obstáculos que os alunos enfrentam, e porque razão acontecem. Encontra-se também no presente documento, informações sobre o produto final, a ferramenta DEIprocessing e o procedimento até este ser alcançado.

Posteriormente, são abordados os testes realizados no produto desenvolvido, bem como os seus resultados e conclusões. Por fim, é feita uma descrição geral e final do trabalho empreendido ao longo destes catorze meses, com perspectivas de um trabalho futuro.

Palavras-Chave

Processing, Tradutor, Compilador, IDE Processing, Ferramentas Tecnológicas, Introdução à Programação, Visualização Interativa, Ensino, Obstáculos e Dificuldades dos Alunos, Auxílio à Programação, Aprendizagem.

Esta página foi propositadamente deixada em branco.

Conteúdo

Esta página foi propositadamente deixada em branco.

Esta página foi propositadamente deixada em branco.

Lista de Figuras

Esta página foi propositadamente deixada em branco.

Lista de Tabelas

Esta página foi propositadamente deixada em branco.

Capítulo 1

Introdução

Este relatório final foi feito no ano lectivo 2021/22, no âmbito da dissertação/estágio, "Visualização interativa de programas escritos em Processing", do aluno Pedro Carvalho, que frequenta o último ano do mestrado de Engenharia Informática, com especialização em Engenharia de Software. O estágio foi realizado no *Cognitive and Media Systems Group* do Centro de Informática e Sistemas da Universidade de Coimbra (CISUC), um centro de investigação e comunicação criado em 1991, com cerca 150 investigadores [1].

Neste capitulo é descrito o enquadramento do projecto (nomeadamente, o que é a linguagem Processing) e as motivações do mesmo, os objetivos e a explicação simplificada de como está estruturado este relatório.

1.1 Enquadramento e Motivações

A linguagem Processing foi criado em 2001, por dois estudantes universitários, Ben Fry e Casey Reas, no laboratório de investigação do MIT Media Lab, integrado no grupo de trabalho de John Maeda, *Aesthetics and Computation* (Estética e Computação). Tratava-se inicialmente de um projeto de investigação, que continuou depois a ser desenvolvido nos seus tempos livres. Em 2012 criaram com Daniel Shiffman a Processing Foundation. Desde o início que o Processing tinha como objetivo ser uma linguagem de programação de iniciação. Por esta razão, a maioria das vezes é lecionado no secundário, em escolas de artes, ou nos primeiros anos de ensino superior, em cursos de artes visuais e licenciaturas relacionadas com a informática [2, 3, 4].

Processing é voltado para a criação de conteúdo visual e interativo, pelo que os primeiros programas começam com o desenho, sendo que os estudantes ficam, normalmente, bastante satisfeitos com a sua aprendizagem, pois em pouco tempo de utilização do software, são capazes de produzir algo visível (o desenho). O facto de isto acontecer permite que esta linguagem de programação seja facilmente integrada em cursos de design, artes e arquitetura, o que por sua vez faz com que o número de estudantes em aulas de áreas de programação aumente. O software do Processing, a linguagem e o IDE, é usado por milhares de designers, artistas e arquitetos para criarem o seu próprio trabalho, sendo também usado para gerar efeitos visuais audio-reativos, vídeos e filmes e ainda para criar projetos de design de palco, para espetáculos de dança e música, etc. Um facto importante sobre o Processing e a sua cultura, é a forma como o software consegue envolver uma nova geração de artistas visuais, que consideram saber programar uma parte essencial da sua prática criativa [2].

Foi com o intuito de envolver e facilitar a vida de novos estudantes na aprendizagem da linguagem de programação Processing, que este estágio foi realizado de forma a desenvolver uma ferramenta que permitisse uma visualização mais interativa dos programas desenvolvidos em Processing. Isto fez com que fosse possível explorar uma parte mais pedagógica da programação, o que foi considerado um desafio motivador extra. Tendo este estágio especificações únicas, fez com que aplicações/ferramentas com características semelhantes fossem escassas, dai a importância do desenvolvimento deste projeto.

1.2 Objetivos

Como mencionado anteriormente, o objetivo principal deste estágio foi o desenvolvimento do suporte à iniciação da aprendizagem em Processing de forma interativa. Este projeto visou a possibilidade de construir e simular programas simples, escritos em Processing, e que estes fossem simulados instrução a instrução, para que os utilizadores possam ver como as diferentes instruções são executadas e produzem determinados resultados.

Um outro objetivo, numa componente mais pedagógica deste projeto, que vale a pena realçar é a intenção de que esta ferramenta possa vir a ser utilizada nas aulas de Introdução à Programação e Resolução de Problemas (IPRP), lecionada no primeiro ano de Licenciatura de Design e Multimédia (LDM), curso da Faculdade de Ciências e Tecnologias da Universidade de Coimbra (FCTUC), com o intuito de facilitar tanto os docentes que lecionam as aulas, agilizando a explicação e demonstração de programas simples escritos em Processing, assim como os alunos, facilitando a interpretação e compreensão do código feito assim como a linguagem em si.

Para que os objetivos acima assinalados fossem cumpridos era necessário que outros mais técnicos fossem definidos e concluídos. A solução escolhida consiste em criar uma sub linguagem de Processing, adaptada do mesmo, e que se consiga auxiliar o utilizador em tempo real, como por exemplo, a simulação passo a passo para melhor perceção do utilizador.

Para tal, foi preciso criar o nosso próprio tradutor para a sub linguagem. Processing tem como base Java, assim como, cada sketch é na realidade uma subclasse da classe de Java: "PApplet", que implementa a maioria das *features* em Processing [5].

De seguida foi feita a análise léxica, a análise sintática e também a tradução orientada pela sintaxe, que é baseada como o próprio nome indica, nas regras sintáticas com o auxilio de uma Abstract Syntax Tree (AST). A partir destas três fases iniciais foi também feita uma terceira chamada tabela de símbolos que permite identificar as variáveis e funções locais e globais.[6]

Com base nestas tarefas supra referidas foi, posteriormente, criada uma ferramenta de Processing que viria a ser a ponte de ligação entre o tradutor e o Integrated Development Environment (IDE) Processing utilizado por eventuais utilizadores, tendo este sido escrito em Java, ao contrário do tradutor que foi escrito em C [7, 8, 9, 10, 11, 12, 13].

1.3 Estrutura do Relatório

Esta secção dedica-se à breve descrição da forma como o documento se encontra estruturalmente organizado.

Estado da Arte - Este capítulo começa com uma análise sobre os diferentes tutoriais

existentes para a linguagem Processing, seguido da análise de dois em específico: "Hello Processing!" e "Getting Started". Depois existe o estudo das ferramentas com características semelhantes às desejáveis. O último tópico dentro do estado da arte reporta-se aos aspetos pedagógicos onde se aborda a importância no ensino das ferramentas tecnológicas, bem como das dificuldades dos alunos na introdução à programação.

Planeamento – Onde é apresentado cronologicamente o trabalho do primeiro semestre e o planeamento, previsto anteriormente, do que se pretendia fazer no segundo e o que realmente foi feito num planeamento reformulado do segundo semestre, recorrendo a diagramas de Gantt.

Desenvolvimento da ferramenta – Como o próprio nome indica é descrito como se chegou ao produto final da ferramenta DEIprocessing, estamos este capítulo dividido em três partes: o Tradutor, o OpenGL e a Ferramenta do IDE Processing.

Descrição da ferramenta – Neste capítulo aborda-se a ferramenta à qual se deu o nome de DEIprocessing, onde se faz uma apresentação desta e da sua interface.

Testes – No capítulo de testes desenvolve-se aquilo a que se chamou de testes iniciais e mostra-se os realizados que asseguram o funcionamento de funções (e não só), assim como os testes de usabilidade realizados juntamente com o seu *feedback*.

Conclusão – Uma visão geral e final do trabalho realizado e perspetiva do trabalho futuro que pode vir a ser desenvolvido.

Esta página foi propositadamente deixada em branco.

Capítulo 2

Estado da Arte

O presente capítulo contém uma análise sobre as diferentes ferramentas/aplicações já existentes, com objetivos semelhantes, assim como *frameworks* e tecnologias adequadas para o desenvolvimento da ferramenta pretendida. Das diferentes ferramentas analisadas, algumas foram utilizadas, outras que se achou importante referir pela sua semelhança com aquilo que é pretendido ou porque ajudaram a retirar ideias para o foco do presente estágio. Envolve ainda a investigação de áreas pedagógicas relacionadas com a programação.

Sendo o foco deste estágio desenvolver uma ferramenta de suporte à aprendizagem da linguagem Processing, torna-se inevitável falar dos tutoriais existentes para esta linguagem de programação e os seus aspetos positivos e negativos, assim como as diferentes abordagens no ensino de linguagens de programação.

Esta análise tem como objetivo determinar a sua atratividade e compreender a evolução das suas oportunidades e ameaças [14]. Foi também possível perceber quais as dificuldades mais comuns para quem inicia a aprendizagem de linguagens de programação e entender que nem sempre os problemas estão do lado do estudante. [15, 16]

Por fim, estudar algumas aplicações e conceitos que, no futuro, poderão integrar e/ou melhorar a ferramenta desenvolvida neste estágio, fazendo com que esta seja mais apelativa e útil.

2.1 Tutoriais para Processing

Existem diversos tutoriais online para Processing, assim como para uma grande parte de linguagens de programação, sendo maioritariamente apresentados sob o formato de blog ou em formato de vídeo. Porém, os primeiros apresentam, na generalidade, duas grandes desvantagens: texto de explicação tendencialmente extenso e predominância de conteúdos estáticos, ou seja, não existe qualquer interação no *browser* e se existe é aquela que é a mais comum em Processing, em que se corre o código e a sua compilação aparece imediatamente desenhada (quando é o caso para aparecer visualmente). As Figuras 2.1 e 2.2 apresentam um exemplo disso. Como amostras de tutoriais em formato de blog temos "Happy Coding" [17], "Fun Programming" [18], o próprio site Processing na secção de "Text Tutorial" [19], entre outros.

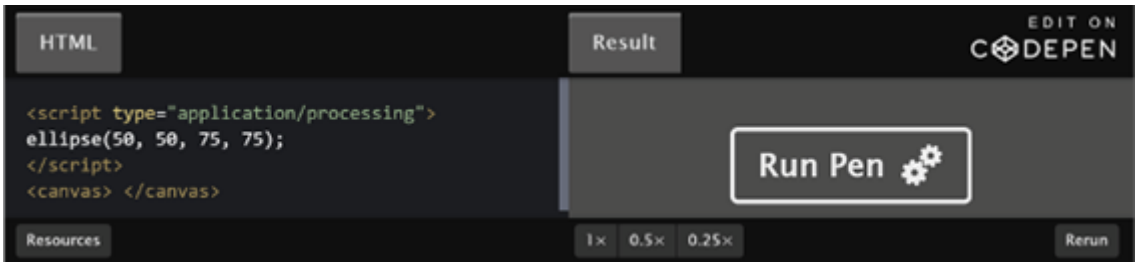


Figura 2.1: Imagem retirada do tutorial Happy Coding antes de carregar no botão "Run Pen"

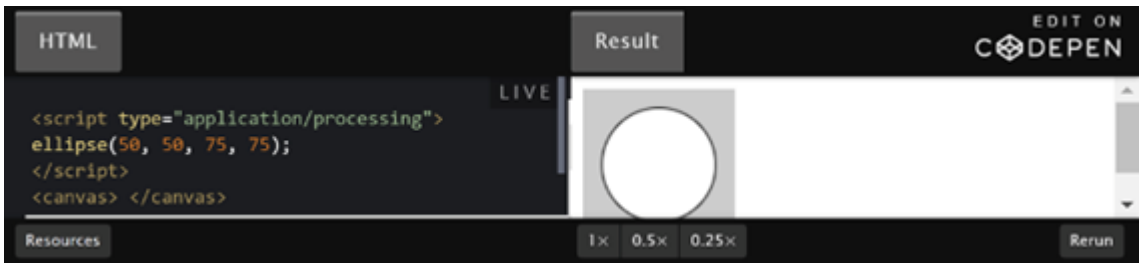


Figura 2.2: Imagem retirada do tutorial Happy Coding depois de carregar no botão "Run Pen"

Alguns dos tutoriais que se destacam pelas suas características dinâmicas e interativas, são o "Hello Processing!" [20] e o "Getting Started" [21], dos quais iremos falar mais nas próximas subsecções.

2.1.1 Hello Processing!

Começando pelo tutorial "Hello Processing!", realça-se a sua dinâmica pelo acompanhamento do vídeo com o aparecimento de código, permitindo ainda alterar o mesmo, bem como o seu output visual, que faz com que seja mais fácil para o utilizador entender e praticar o que acabou de lhe ser ensinado. Esta foi a principal razão pela qual se achou importante referir este tutorial [20]. Pode-se ver um exemplo disso na Figura 2.4. Foi produzido pela Processing Foundation e lançado em dezembro de 2013 [22]. Tem a duração de cerca de uma hora e está dividido em seis vídeos representados pelas secções representadas na Figura 2.3.

Figura 2.3: Header do tutorial "Hello Processing!" que permite escolher o vídeo

Sendo que cada vídeo representa o seguinte:

- "Hello" -> que faz uma breve introdução do que é a linguagem de programação Processing e onde é usado.

- "Shapes" -> neste segundo vídeo são dadas duas funções de formas: o retângulo ("rect()") e a elipse ("ellipse()") e as suas respetivas coordenadas e tamanhos.

- "Color" -> aqui são abordados temas relacionados com a "cor" das formas, do contorno, e do fundo a partir das funções "stroke()", "fill()" e "background()".

- "Interact" -> muito sucintamente, este vídeo fala sobre as animações e as funções "setup()" e "draw()".

- "Questions" -> por fim, este último introduz estruturas de decisão (os "ifs" e o "elses").

- "GoodBye" -> o sexto vídeo é praticamente uma despedida e já não tem novas informações para o tutorial.

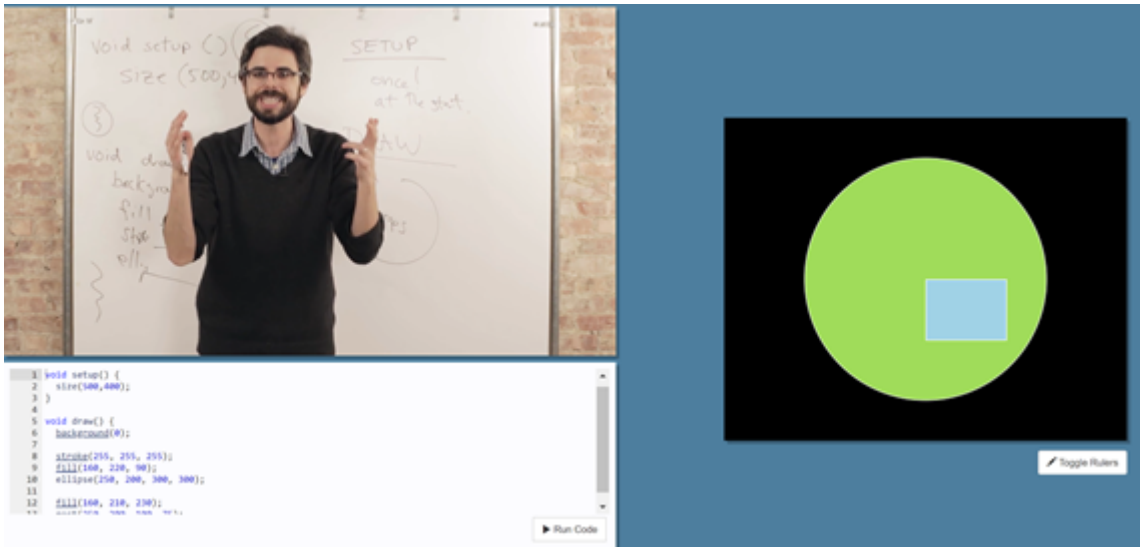


Figura 2.4: Interface do utilizador do tutorial "Hello Processing!"

2.1.2 Getting Started

A ferramenta "Getting Started" foi criada por Jae Hyun e é baseada no tutorial "Getting Started" do site Processing [23]. Esta versão melhorada tem uma aprendizagem mais faseada, assim como uma interface de acesso facilitado por ser disponibilizado no Integrated Development Environment (IDE) Processing.

Este tutorial é uma das ferramentas chamadas de "Contributed tools", que são desenvolvidas por membros da comunidade Processing e o seu download pode ser feito diretamente no IDE do Processing na opção "Add Tool..." [24]. Pode-se ver as primeiras fases desta tutorial na Figura 2.5.

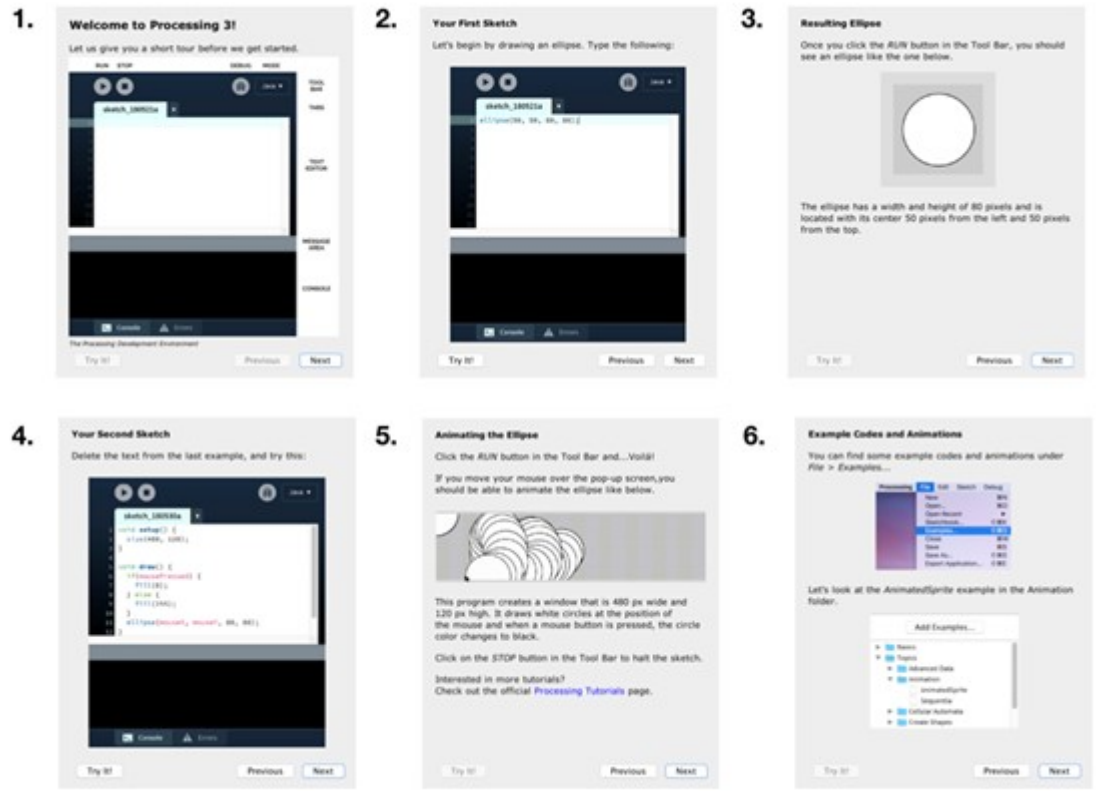


Figura 2.5: Prints das primeiras fases do tutorial "Getting Started"

2.2 Ferramentas

O propósito desta secção das ferramentas é ajudar a atingir um objetivo e/ou facilitar a conclusão do processo. Por isso, foi necessário um estudo das ferramentas tecnológicas disponíveis com o intuito de identificar as características desejáveis numa ferramenta que tem como propósito principal auxiliar o ensino da programação, recorrendo à visualização interativa de programas simples escritos em Processing.

Nesta sub secção refere-se a existência da *framework* turtle, porque apesar de ter como base a linguagem de programção Python é, como a linguagem Processing, muitas vezes usado para a introdução à programação recorrendo à visualização gráfica. Uma grande vantagem que o turtle tem sobre a linguagem Processing é conseguimos ver os gráficos a serem desenhados de forma progressiva até ao estado final do output, enquanto em Processing os gráficos são logo todos desenhados e só vemos o output no estado final. Como tal não conseguimos ver o seu processo. Uma das limitações do *turtle* é não ser tão focado em variáveis, que é um conceito importante [25, 26].

Foram também avaliadas ferramentas já existentes, semelhantes àquilo que se pretendia e/ou com características semelhantes, sendo o exemplo disso os seguintes:

visualgo permite a visualização de estrutura de dados e algoritmos através de animações, ou seja, a execução de algoritmos ou estrutura de dados (como na Figura 2.6) de instrução em instrução acompanhada de uma ilustração. Quando este passa para

a próxima linha de código (instrução), ocorre em simultâneo a animação do movimento referente a essa nova linha, e assim sucessivamente até ao final do programa, agilizando a aprendizagem e compreensão destes algoritmos e estrutura de dados. O visualgo foi criado por Steven Harlim em 2011 [27, 28].

Figura 2.6: Exemplo de utilização do visualgo

Interactive animations (anim) -> uma versão mais simples, mas com a mesma base que o visualgo, para aprender algoritmos e programação através de animações interativas e, mais uma vez, capaz de executar de instrução a instrução em simultâneo com a animação. É possível ainda pausar, recomeçar e alterar a velocidade da animação. Pode-se ver um exemplo disso na Figura 2.7. Esta ferramenta está disponível em inglês, húngaro e eslovaco. Foi criada em 2012 pelo eslovaco Ladislav Végh [29, 30]. A ferramenta está dividida em cinco secções: "Basic algorithms" [30], "Sorting cards" [31], "Sorting algorithms 1" [32], "Sorting algorithms 2" [33] e "River crossings" [34].

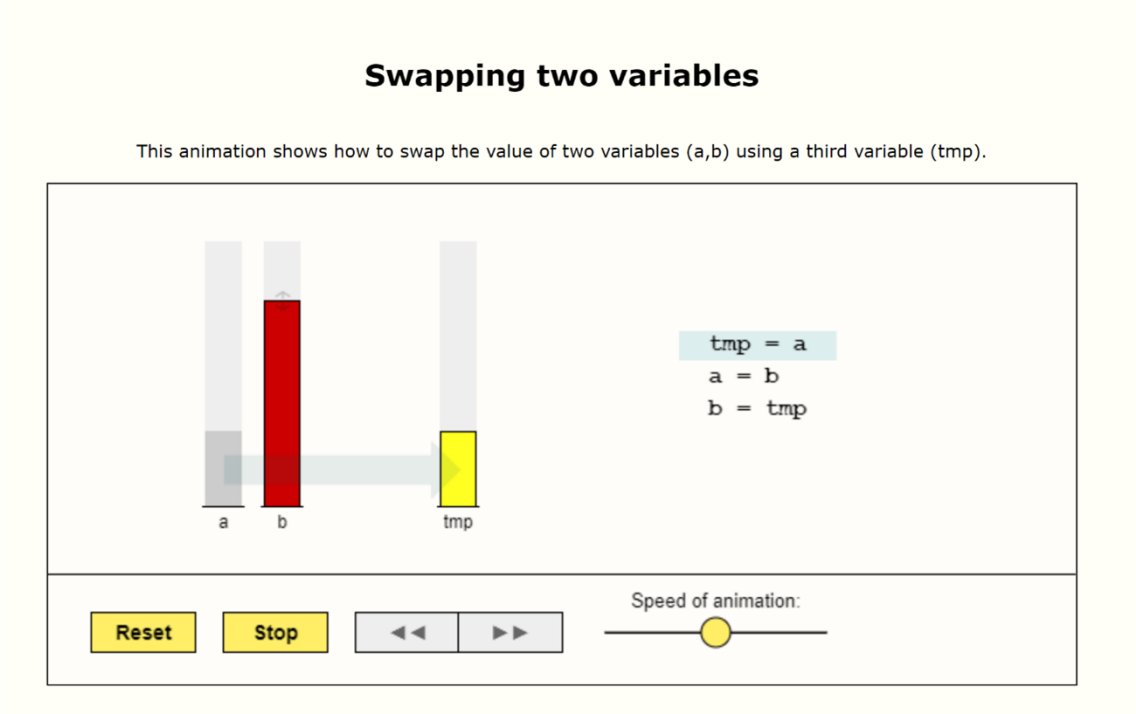


Figura 2.7: Exemplo do anim (trocar de duas variáveis)

A Tabela 2.1 representa uma comparação de funcionalidades e atributos, entre as duas ferramentas já referidas (visualgo e anim) e a ferramenta produto deste projeto, DEIprocessing.

Tabela 2.1: Tabela de comparação de funcionalidades e atributos entre Visualgo, Anim e DEIprocessing.

	Desenho interativo	Realce da linha em execução	Iteração por frames	Exemplos	Webpage	Dar para utilizar código do próprio utilizador
Visualgo	**X**	**X**	**X**	**X**	**X**	-
Anim	**X**	**X**	**X**	**X**	**X**	-
DEIprocessing	**X**	**X**	**X**	**X**	-	**X**

2.3 Aspetos pedagógicos

Esta dissertação envolveu uma componente pedagógica que adicionou complexidade ao trabalho, ainda que seja bastante interessante.

Processing faz com que seja possível a introdução de conceitos de software no contexto de *media arts*. É um *environment* excelente para iniciantes, uma vez que existem resultados visuais imediatos. Processing ajuda pessoas dificuldades através de estruturas de programação concisas, sintaxe familiar, exemplos claros e bibliotecas adicionais [35].

O contacto com os alunos e professores era indispensável para que se tivesse uma melhor noção da realidade procurada. Estamos a falar de uma ferramenta de suporte à aprendizagem e este tema é uma investigação, debate e problema que têm causado dificuldades a muitos especialistas.

Um tópico considerado desafiante na introdução à programação é a criação de ferramentas que suportem a aprendizagem de diferentes tipos de projetos mas de forma a não abdicar de nenhuma abordagem de ensino tradicional essencial, que garanta que é possível a reutilização da ferramenta nestas situações distintas [36].

Ferramentas e demonstrações interativas são amplamente utilizadas para ensinar e reforçar conceitos em várias áreas de informática, como algoritmos, estruturas de dados e arquitetura [37, 38, 39, 40].

A aparência das ferramentas interativas que auxiliam a aprendizagem à programação é importante, na medida em que havia uma forte propensão para os alunos "irem diretos ao assunto" na interação com a ferramenta gráfica imediatamente e ignorar as instruções e exemplos que as acompanham [40].

O reconhecimento das aplicações criativas de programação levou ao desenvolvimento de sistemas de codificação criativos: linguagens textuais como Processing [41] e p5.js [42]. Essas linguagens fornecem aos artistas abstrações específicas de domínio para gerar e transformar imagens, vídeo, geometria e estilo [43], ao mesmo tempo que retêm o acesso a operações de programação gerais, como *loops*, listas, condicionais e funções [44]. A combinação de abstrações orientadas para a arte e programação em geral torna as ferramentas de codificação criativas extremamente expressivas [45].

A de utilização de ferramentas tecnológicas em vez de técnicas convencionais, tem vindo a ser cada vez mais popular no século 21, seja em escolas ou universidade que abrangem a área da programação, inclusive a maioria dos investigadores concorda no beneficio do uso de ferramentas tecnológicas nos diferentes sistemas de educação. [46]

O ensino com a ajuda destas ferramentas tem-se mostrado, na generalidade, mais efetivo ou pelo menos tão efetivo como os métodos tradicionais de ensino. O propósito do uso da tecnologia no processo de ensino e de aprendizagem passa por melhorar a produtividade, a eficácia das práticas atuais e trazer mudanças pedagógicas benéficas para a melhoria da educação. Estas ferramentas pretendem também criar um ambiente inovador para estudantes [46].

O papel da tecnologia no ensino é muito significativo, pois o uso de ferramentas tecnológicas melhora a qualidade da educação. Aprender e ensinar com o auxílio de ferramentas tecnológicas é uma forma organizada de conceituar a execução e avaliação do sistema educacional. [47]

Numa aula com um ambiente convencional, os alunos apenas prestam atenção ao professor sem qualquer tipo de contacto, o que facilmente pode fazer com que os primeiros fiquem cansados e aborrecidos [48]. O uso de ferramentas tecnológicas inovadoras na educação e aprendizagem leva a estudos mais eficientes e faz com que os estudantes tenham a possibilidade de entender melhor com uma atmosfera baseada na tecnologia.

A forma como os professores conduzem a aula pode ser um fator que leva a que o aluno não se interesse pela mesma. Alunos diferentes têm motivações diferentes [16].

De acordo com [46], os estudantes que foram ensinados pela implementação de ferramentas tecnológicas mostraram mais capacidade de competência e capacidade de análise crítica e cognitiva dos problemas quando comparados a estudantes que passaram pelo ensino

tradicional. Gráficos animados, simulações de áudio e vídeo incorporados de forma organizada realizam o processo de aprendizagem de novos conhecimentos com muito mais eficiência. As ferramentas tecnológicas apenas têm um papel importante no ambiente académico se estimularem a missão do ensino, que é socializar, fornecer instrução e educação [40, 49, 50, 51].

A introdução à aprendizagem de uma linguagem de programação por si só é o motivo que a torna o difícil que outros temas pois é algo novo para a maioria dos alunos [16], mesmo assim, existem diversos fatores que contribuem para os erros e outras dificuldades que os estudantes enfrentam. Entre muitos fatores, estão a complexidade da tarefa e a carga cognitiva, uma vez que os iniciantes que estão a aprender a programar ainda não estão familiarizados com todos os novos termos e sintaxe da linguagem de programação. Por isso podem, por vezes, esquecer-se de algumas partes sintáticas básicas dos *statements* como parênteses, chavetas e pontos e vírgulas ou podem confundir operadores básicos ao escrevem programas. Quando as tarefas se tornam mais complicadas os estudantes tendem a fazer mais erros sintáticos [15].

Outro fator que pode gerar dificuldades aos alunos é o conhecimento matemático. Por exemplo, os estudantes podem confundir atribuições de variáveis num programa com expressões algébricas, que parecem semelhantes mas significam algo completamente diferente. Como os alunos, à priori, tiveram anos de aulas de matemática, as notações usadas em programação, quando confundidas com matemática convencional, podem criar obstáculos à compreensão cognitiva [15].

Fatores relacionados com o *environment* podem também causar algumas barreiras. A ambiguidade traz flexibilidade aos programadores profissionais mas também leva confusão e desafios para os que estão a aprender. As mensagens de erro enigmáticas e não informativas do compilador também tornam a correção de erros mais difícil para os iniciantes como por exemplo, a mensagem de erro "cannot find symbol" em Java [15].

Em suma, as várias fontes de problemas estão, muitas vezes, relacionadas com o conhecimento e a experiência passados do estudante, dai que para entender estas dificuldades e razão da sua causa é necessário ter a noção desse conhecimento anterior dos alunos. Portanto, é vital desenvolver o conteúdo pedagógico e conhecer os erros e dificuldades dos estudantes e por isso foi feito um formulário ainda no primeiro semestre com este objetivo, onde foi possível concluir que alunos com muito pouco conhecimento em Processing cometiam erros muito básicos (disponível no apêndice A, 7.1) [52]. Como tal, é essencial aplicar estratégicas e ferramentas de ensino como aquela que se desenvolveu nesta dissertação [15].

Esta página foi propositadamente deixada em branco.

Capítulo 3

Planeamento

Neste capítulo aborda-se o planeamento previamente definido no primeiro semestre. Para a entrega intermédia e que mudanças sofreu após o surgimento de alguns obstáculos, dificuldades e mudanças de decisão. De referir também que ao longo de toda esta dissertação foram feitos relatórios 15-5 semanais com o objetivo de relatar aquilo que se tinha realizado na semana anterior e o que se planeava fazer na semana que se seguia. Foram também feitas reuniões quinzenais ao longo dos trabalhos.

3.1 Planeamento intermédio

Este planeamento é referente ao espaço temporal do primeiro semestre que decorreu desde setembro de 2020 a janeiro de 2021.

3.1.1 Primeiro Semestre

A apresentação do diagrama de Gantt na Figura 3.1 demonstra o trabalho realizado durante o primeiro semestre. Nos primeiros 12 dias teve lugar a "familiarização com o Processing" (tendo o estágio começado no dia 25 de setembro). Esta primeira etapa consistiu muito na aprendizagem da linguagem em causa, pois nunca tinha tido contacto com a mesma, através de exercícios simples, assim como, de tutoriais já referidos no estado da arte.

Seguiu-se o "estudo do estado da Arte" que teve a duração de 24 dias baseado praticamente na investigação e descoberta de novos conceitos, métodos e tecnologias que nos permitissem avaliar o dito "estado da arte".

Durante 11 dias verificou-se a "adaptação com as tecnologias a usar", o que coincidiu com o já mencionado "estudo do estado da Arte" e com a tarefa seguinte: "desenvolvimento do compilador (ficheiros lex e bison)" que teve início no dia 16 de novembro e que decorreu até ao dia 10 de dezembro do ano passado. Estas duas fases "adaptação com as tecnologias a usar" e "desenvolvimento do compilador (ficheiros lex e bison)" foi onde se começou a implementar a prática, recolhendo já alguns conhecimentos transmitidos em cadeiras do curso de Licenciatura em Engenharia Informática, neste caso a cadeira de compiladores, tendo sido escrita grande parte da nossa gramática.

Por fim, foi elaborado o relatório intermédio, que levou 26 dias a ser concluído para que fosse possível apresentar o trabalho realizado no primeiro semestre e o que se havia planeado no segundo.

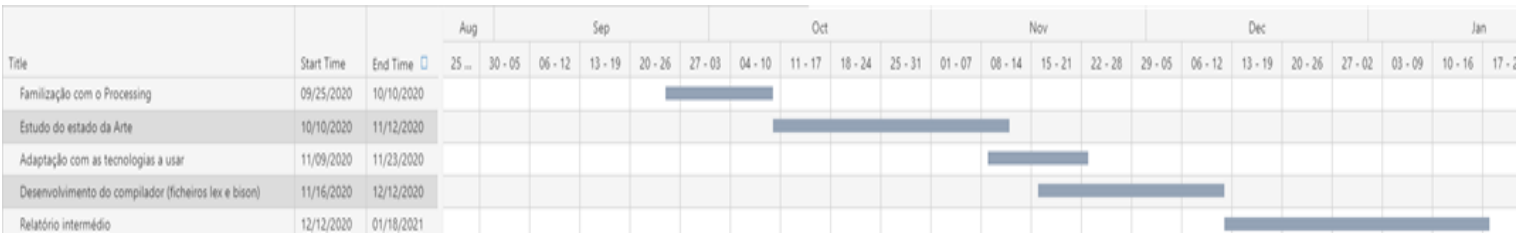


Figura 3.1: Diagrama de Gantt do primeiro semestre

3.1.2 Segundo Semestre

O segundo semestre deu continuidade ao desenvolvimento daquilo que agora chamamos de tradutor iniciado no primeiro semestre, com a tradução orientada pela sintaxe, a análise semântica e possíveis alterações/atualizações nos ficheiros lex e bison. A seguir será ainda necessário fazer a otimização do código e a geração de código.

Estas últimas fases da criação do que chamávamos, anteriormente, de compilador coincidirão com o desenvolvimento das funcionalidades principais definidas anteriormente: visualização da correlação código-output e os exemplos organizados. Depois destas tarefas concluídas e se se achar que haverá tempo suficiente, passarei à execução das tarefas secundárias: desenho interativo e *reference* interativo. Caso contrário, das funcionalidades principais passarei diretamente para o teste de usabilidade e consequentes refinamentos à ferramenta, posteriormente ocorrerá a escrita do artigo científico sobre o projeto e por fim, a escrita da dissertação final.

Como se trata de um planeamento só podemos fazer estimativas, para que estas sejam as mais aproximadas da realidade possível, sendo a métrica usada será semanas:

- Tradução orientada pela sintaxe: 2 semanas
- Análise semântica: 2 semanas
- Alterações/atualizações ficheiros lex e bison: 1 semana
- Otimização do código: 2 semanas
- Geração de código: 2 semanas
- Visualização da correlação código-input: 2 semanas
- Exemplos organizados: 2 semanas
- Desenho interativo: 1 semana
- Reference interativo: 1 semana
- Teste de usabilidade: 1 semana
- Refinamentos da ferramenta: 1 semana
- Escrita do artigo científico sobre o projeto: 4 semanas
- Escrita da dissertação final: 4 semanas

Este planeamento foi feito no pressuposto de que o segundo semestre começa no dia 1 de fevereiro, primeiro dia útil após a defesa da primeira parte da dissertação e que termina no dia 30 de junho, dia da entrega final da dissertação. Outra nota a acrescentar é que apesar de as funcionalidades secundárias se encontrarem estimadas, tratava-se apenas de uma visão temporal otimista de forma a que caso existisse mais tempo para realizar tarefas extra, já estariam planeadas.

Tudo isto é representado através de um diagrama de Gantt, na Figura 3.2.

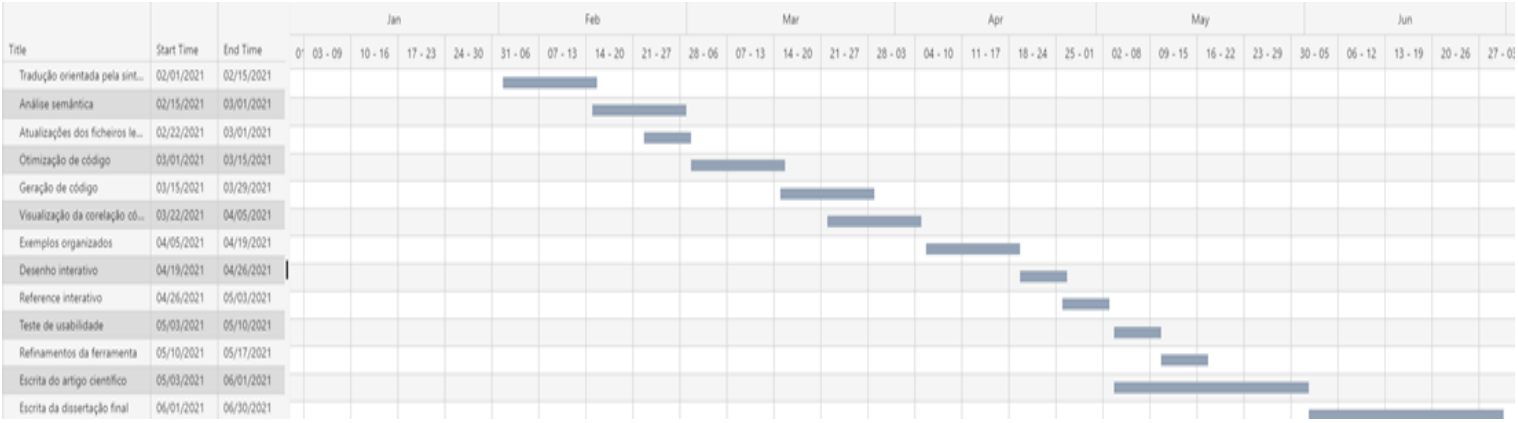


Figura 3.2: Diagrama de Gantt do segundo semestre

3.2 Planeamento finalizado

Ao contrário do planeamento anteriormente descrito, este já reflete o estado final da dissertação. Esta secção servirá para explicar o que foi alterado e o porque de ter sido alterado em relação ao planeamento intermédio e também como foi organizado este segundo semestre face as dificuldades e soluções encontradas. À priori podemos ter em conta que o planeamento anterior foi feito com vista a terminar em junho enquanto esta dissertação foi feita até ao final de Outubro.

3.2.1 Alterações e reformulações

Como já referido em cima é importante dizer o que mudou e porque mudou. E a primeira, e creio que mais importante, é a questão temporal, uma vez que a dissertação foi estendida por mais 4 meses do que aquilo que era inicialmente previsto.

Esta mudança deve-se muito aos aspectos de engenharia e desenvolvimento (estas são as razões de outras alterações no planeamento também), por isso era necessário mais tempo para que fosse possível a criação de uma ferramenta mais apelativa e interativa. E dai que tivesse havido a necessidade do alargamento do prazo de entrega. Em relação a questão dos testes com os alunos, aqueles que estariam mais acessíveis e aptos a testar a ferramenta desenvolvida, eram aqueles que entraram apenas em Setembro para o curso de Licenciatura de Design e Multimédia (LDM) e que frequentam a cadeira de Introdução à Programação e Resolução de Problemas (IPRP), que com este adiamento permitiu que fizessem alguns testes .

As etapas inicialmente descritas como “Otimização de código” e “Geração de código” foram

colocadas de parte, pois para aquilo que se pretendia com o nosso tradutor existiam outras tarefas mais importantes que consumiram mais tempo. Um exemplo disto foi a questão do "desenho interativo", ao contrário do que se tinha planeado inicialmente achou-se que esta característica levaria menos tempo a ser concluída para a nossa ferramenta e como tal teria de haver uma mudança no planeamento. Para que isso acontecesse era necessário que o nosso tradutor fizesse a tradução da gramática já criada no primeiro semestre para OpenGL e esta tradução foi uma tradução orientada pela sintaxe através de uma Abstract Syntax Tree (AST).

Ainda sobre as alterações mais relevantes, foi necessário tempo para a aprendizagem da utilização do OpenGL e bibliotecas adjacentes, assim como tempo para aprender a trabalhar com as ferramentas do IDE Processing em si. Como tal, apenas a tarefa secundária "*reference* interativo" foi deixada de lado para que houvesse mais tempo deixado para outras, como o *highlight* daquilo que muda consoante a linha de código em execução, para que fosse possível uma melhor compreensão do programa.

Por último, a tarefa que não foi realizada foi a "escrita do artigo científico sobre o projeto".

3.2.2 Reformulação do Segundo Semestre

Esta sub secção destina-se a descrever como o segundo semestre foi dividido pelas tarefas e eventuais dificuldades. Por uma questão de coerência a medida usada nesta explicação e também no diagrama de Gantt 3.3, foi semanas, tal como tinha sido feito anteriormente no relatório intermédio.

Começando o semestre por terminar a nossa AST que nos permitia realizar a tradução orientada pela sintaxe, passamos à análise semântica, bem como, a alterações que eram necessárias nos ficheiros de lex e bison à medida que o projeto avançava. Posto isto, foi necessário o estudo e a aprendizagem de aquilo que acabou por ser a nossa solução no que diz respeito à representação dos gráficos/desenhos, falamos da aprendizagem e utilização do OpenGL. Esta tarefa pode-se dizer que foi realizada em simultâneo com aquelas que chamamos de funcionalidades como a visualização da correlação código-input, o desenho interativo e os *highlights* da linha de código, posteriormente os exemplos organizados. Após a conclusão desta tarefas foi necessário, mais uma vez, o estudo e o desenvolvimento da ferramenta do IDE Processing e uma vez terminada passou-se para os testes de utilização, que consoante o *feedback* foi necessário os ajustes/refinamentos da ferramenta. Por fim procedeu-se à escrita da dissertação final.

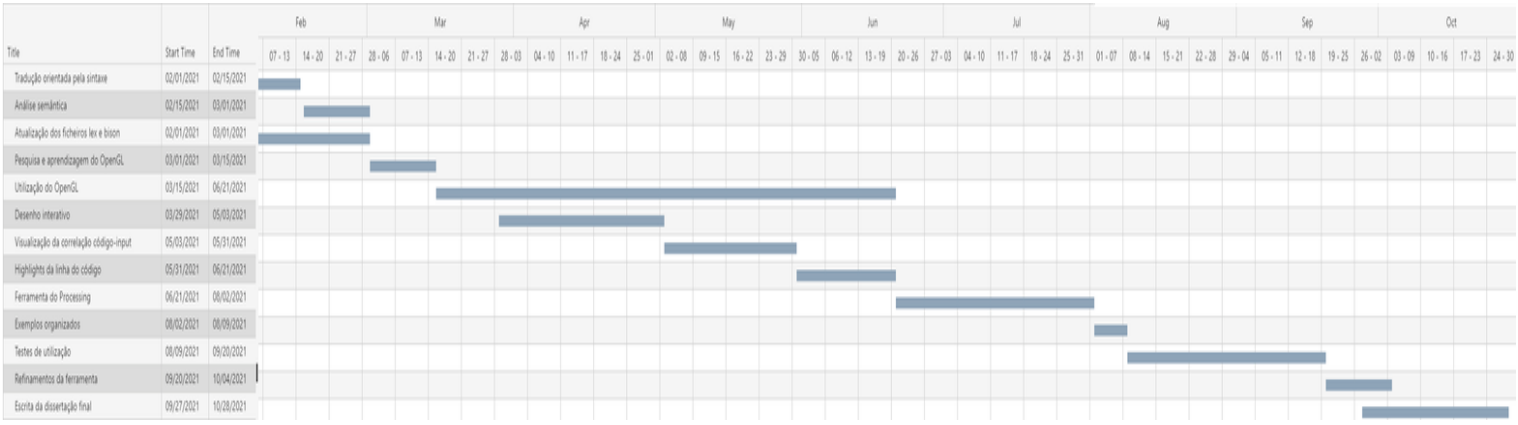


Figura 3.3: Diagrama de Gantt do segundo semestre reformulado

Como tal, o tempo, em semanas, utilizado por tarefa foi o seguinte:

- Tradução orientada pela sintaxe: 2 semanas
- Análise semântica: 2 semanas
- Alterações/atualizações ficheiros lex e bison: 4 semanas
- Pesquisa e aprendizagem do OpenGL: 2 semanas
- Utilização do OpenGL: 14 semanas
- Desenho interativo: 5 semanas
- Visualização da correlação código-input: 4 semanas
- *Highlights* da linha do código: 3 semanas
- Ferramenta do Processing: 6 semanas
- Exemplos organizados: 1 semana
- Teste de usabilidade: 6 semanas
- Refinamentos da ferramenta: 2 semanas
- Escrita da dissertação final: 4 semanas

Umas notas finais sobre este tema: o facto de algumas destas tarefas se realizarem em simultâneo/intercaladas e que a discrepância do tempo estimado anteriormente para o que realmente aconteceu, deve-se a maioritariamente à falta de noção do quanto complexa era na realidade a tarefa e também pela dificuldade em prever tarefas que nunca tinham sido realizadas antes por mim.

Esta página foi propositadamente deixada em branco.

Capítulo 4

Desenvolvimento da ferramenta

Uma das opções estudadas para o desenvolvimento desta dissertação foi a criação de raiz de uma biblioteca em Processing. Pareceu ser uma opção satisfatória, uma vez que a sua implementação, apesar de trabalhosa, não seria impossível. Porém, após uma maior investigação chegou-se à conclusão de que não seria a melhor abordagem e que me iria limitar no principal objetivo de visualizar os programas interativamente [53, 54], até porque, como já referido, a base de Processing é na realidade uma classe de Java chamada Papplet [55, 56, 57, 58].

Como tal, neste capítulo apresenta-se o processo pelo qual se considerou mais adequado, a criação da ferramenta, assim como as diferentes fases tecnologias usadas ao longo da dissertação. O capítulo está dividido em 3 secções: Tradutor, OpenGL e Ferramenta do IDE Processing com o nome de DEIprocessing.

4.1 Tradutor

A primeira fase prática propriamente dita foi o desenvolvimento do tradutor. Pretendia ter o controlo total da gramática, para que a nossa ferramenta tivesse as funcionalidades e características pretendidas e como tal a criação do tradutor foi a solução encontrada. Esta fase começou ainda no primeiro semestre com a evolução das análises lexical e sintática que deram origem aos ficheiros Lex e Bison, respetivamente. Posteriormente foi feita a tradução orientada pela sintaxe através de uma Abstract Syntax Tree (AST) e parte da análise semântica, foi a partir destas ultimas etapas que foi feita a ligação à segunda fase da ferramenta, a Utilização do OpenGL.

4.1.1 Análise léxica e sintática

Na parte inicial do desenvolvimento do nosso tradutor era necessário a implementação de analisadores léxicos e de analisadores sintácticos, o que foi feito com auxilio de ferramentas. Neste caso foi utilizado o Lex, desenvolvido pela Bell [59], e Bison, desenvolvido pela GNU [60], e ambos geram programas codificados na linguagem de programação C [6].

Como tal, a análise léxica é responsável pela leitura sequencial dos caracteres que formam o texto fonte, pela sua separação em palavras, pelo reconhecimento dos símbolos terminais representados por cada palavra. De seguida foi feita a análise sintáctica, que é encarregue de agrupar os símbolos terminais, verificando se formam uma frase sintaticamente correta,

composta de acordo com as regras sintáticas da linguagem.

O Lex é uma ferramenta que gera um analisador léxico a partir de um ficheiro de especificações, ficheiro esse que é dividido em três partes. Na primeira parte são descritas as definições, onde se define os macros (utilizados nas linguagens de programação com vista a facilitar a inserção de instruções repetidas) e se importa os *header files* , a segunda é dedicada às regras de emparelhamento, onde se associa padrões de expressões regulares com instruções em C, e a terceira é reservada às rotinas de utilizador, que contem instruções e funções em C que são literalmente copiados para o *source file* gerado. As partes são delimitadas pelo indicador %% [6, 61]. Pode-se ver um exemplo disto na Figura 4.1.

O Bison é uma ferramenta que gera um analisador sintáctico de arquitectura ascendente LALR(1), por *default*, a partir de um ficheiro de especificações com um formato idêntico ao do Lex. O analisador gerado lê as sequências de *tokens* e decide se estão em conformidade com a sintaxe especificada pela gramática. O Bison é compatível com o Yacc e qualquer pessoa familiarizada com este último deve conseguir usar o primeiro sem qualquer problema, pode ainda gerar código para C, C++ e Java [6, 62, 63].

```
Definition section
%%
Rules section
%%
C code section
```

Figura 4.1: Formato de um ficheiro de especificações Lex (semelhante a um Bison)

Para a compilação dos ficheiros Lex (begin.l) e Bison (begin.y) usamos os seguintes comandos:

- flex begin.l
- bison -dy begin.y

Que resultam nos ficheiros "lex.yy.c" e "y.tab.c", que são os reconhecedores léxicos e sintáticos que serão usados posteriormente.

4.1.2 Abstract Syntax Tree(AST)

Neste caso, a criação de uma AST foi feita com o intuito de se realizar a tradução orientada pela sintaxe, isto é, proceder à tradução baseada nas regras sintáticas previamente definidas. Na tradução orientada pela sintaxe, o reconhecimento do texto de entrada é efectuado apenas sobre a gramática de entrada, enquanto o texto de saída é gerado por execução das ações expressas na gramática tradutora (gramática independente de contexto, na qual os símbolos terminais são divididos em dois conjuntos, símbolos de entrada e símbolos de ações) [6].

Uma AST, como o nome indica, é uma representação em árvore da estrutura sintática do texto (normalmente código-fonte), escrito numa linguagem formal, onde cada nó da árvore denota uma construção que ocorre no texto. São estruturas de dados bastante usadas

em compiladores, assim como em tradutores, para representar a estrutura do código do programa. As ASTs são geralmente o resultado, como neste caso, da analise sintática de um compilador/tradutor. Frequentemente oferecem uma representa intermédia de um programa. Um exemplo de uma AST pode ser encontrado na Figura 4.2 [64].

```
Start
--Begin
----Void draw
----VarDecl
------Color
------Id c1
------ColorLit
--------Virgula
----------Virgula
------------Intlit 200
------------Intlit 200
----------Intlit 7
----VarDecl
------Int
------Id a
------Intlit 30
----If
------Diferente
--------Id a
--------Intlit 20
------Bloco
--------MethodDecl
----------Id fill
----------Id c1
--------MethodDecl
----------Id rect
----------Virgula
------------Virgula
--------------Virgula
----------------Intlit 10
----------------Id a
--------------Intlit 300
------------Intlit 300
------Bloco
--Begin
----Void setup
----MethodDecl
------Id size
------Virgula
--------Intlit 500
--------Intlit 500
----MethodDecl
------Id background
------Virgula
--------Virgula
----------Intlit 255
----------Intlit 0
--------Intlit 0
```

Figura 4.2: Um exemplo de uma AST de um programa simples escrito em Processing

4.1.3 Análise semântica

A análise semântica é a última fase típica de um compilador, que foi feita dentro deste projeto. Esta análise tem como intuito verificar os erros semânticos no *source-code* e recolher as informações necessárias também do *source-code* [6, 65].

Apesar de ainda ter sido desenvolvida uma boa parte desta fase de um compilador, nomeadamente um ficheiro próprio (TabelaDeSimbolos.c e respetivo *header*, TabelaDeSimbolos.h), escrito em C, que permite a criação de uma tabela de identificadores que contêm as funções globais e locais, tal como as variáveis globais e locais, como se pode ver no exemplo da Figura 4.3, esta análise semântica acabou por não ser muito utilizada nem mais desenvolvida, pois entendeu-se que não representava uma parte significativa para o tradutor final [6, 65].

A única utilização direta que acabou por existir foi a inserção das variáveis num ponteiro, que serve como referência para o tradutor, tendo-se aproveitado o código já escrito previamente.

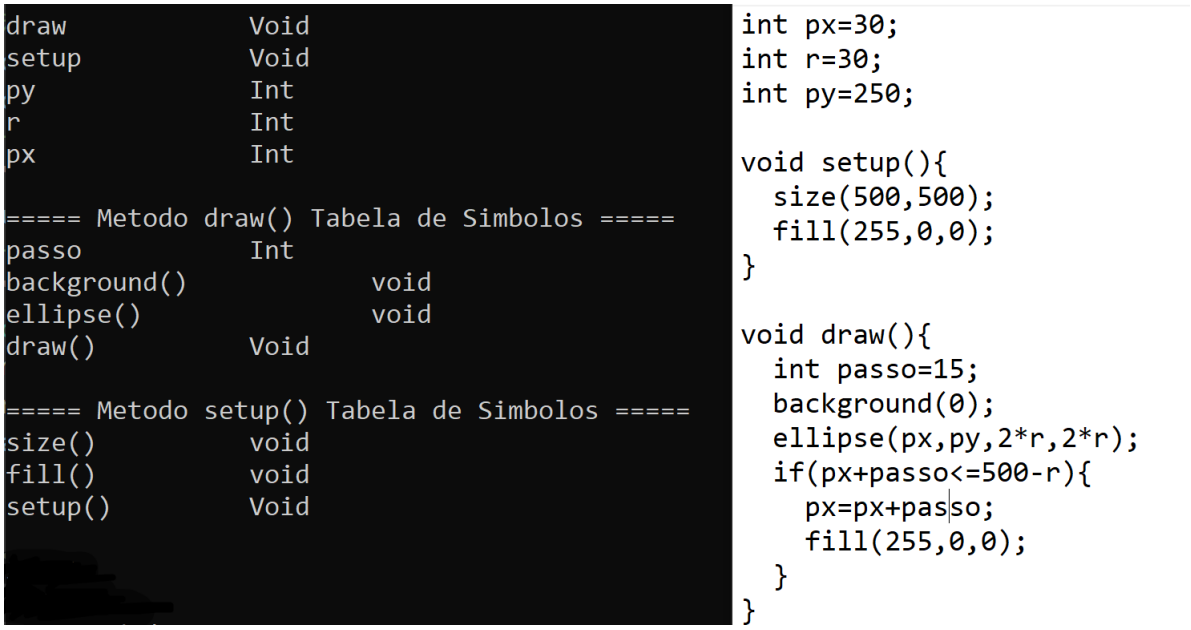


Figura 4.3: Um exemplo da tabela identificadora do código em Processing à direita

4.2 OpenGL

Na medida em que fosse possível a criação de uma ferramenta interativa e que representasse as formas e desenhos da forma mais semelhante possível ao Processing optou-se por fazer a tradução para Open Graphics Library (OpenGL), com o auxilio das bibliotecas *FreeGLUT* e *GLEW* [66, 67].

O OpenGL é uma Applicantion Programming Interface (API) de *cross-platform* para a renderização de gráficos vetoriais em 2D e 3D, gráficos estes que são imagens de computação gráfica definidas por pontos (diferente do turtle em pyhton, referido em cima) em plano Cartesiano. Foi lançado em 1992 e em 2006 passou a ser gerido pela associação, não lucrativa, tecnológica do grupo Khronos [66, 68].

Por fim, o *FreeGLUT* é uma alternativa *open-source* à biblioteca OpengGL Utility Toolkit (GLUT) que permite ao utilizador criar e controlar janelas que com contexto OpenGL numa ampla gama de plataformas e também ler as funções de rato, teclado e joystick [69]. O *GLEW* é uma biblioteca *cross-platform* C/C++ que ajuda na consulta e carregamento de extensões de OpenGL, fornece ainda mecanismos eficientes de tempo de execução que determinam quais extensões OpenGL são suportadas na plataforma de destino [66, 67].

4.2.1 Utilização

Foi a partir da AST que a ligação entre a primeira fase e a segunda é feita. Este processo é realizado percorrendo a árvore criada e consoante o nó da árvore é feita a ação respetiva. As funções traduzidas para o OpenGL foram: ellipse(), rect(), triangle(), background(), fill(), stroke(), noStroke(), size(), draw() e setup(). Para além destes o tradutor suporta a definição de variáveis, if's e for's em Processing. A Figura 4.4 mostra um exemplo de código de uma função.

```
glBegin(GL_QUADS);
    glVertex2f(x,y);
    glVertex2f(x+largura,y);
    glVertex2f(x+largura,y+altura);
    glVertex2f(x,y+altura);
glEnd();

if(strokeFlag == 1){
    glColor3f(strokeRed/255, strokeGreen/255, strokeBlue/255);
    glLineWidth(defaultLineWidth);

    glBegin(GL_LINES);
        glVertex2f(x,y);
        glVertex2f(x+largura,y);
        glVertex2f(x+largura,y);
        glVertex2f(x+largura,y+altura);
        glVertex2f(x+largura,y+altura);
        glVertex2f(x,y+altura);
        glVertex2f(x,y+altura);
        glVertex2f(x,y);
    glEnd();
}
```

Figura 4.4: Exemplo de como um das funções (rect()) é implementada em OpenGL

O OpenGL foi ainda usado para a criação de dois *viewports* para além da principal, um com o intuito de realçar a linha em execução e outro com o objetivo de realçar o que altera consoante a linha em execução, bem como as variáveis definidas e as do sistema. Este tema será mais aprofundado no próximo capitulo.

4.3 Ferramenta do Processing

Para terminar este capitulo, falemos das ferramentas do IDE Processing. Existem dois tipos de ferramentas: *core tools (Auto Format, Color Selector, Create Font)* que são parte da distribuição do Processing e as *contributed tools* cujos desenvolvimento, manutenção e direitos pertencem aos membros da comunidade do Processing. As ferramentas fazem o IDE mais útil e extensível [7].

Neste caso, começámos por tentar que isto acontecesse mas rapidamente se entendeu que a ferramenta do IDE Processing, propriamente dita, serviria quase exclusivamente como uma ponte de ligação entre o nosso tradutor (a partir do executável gerado) e o IDE Procesing, na Figura 4.5 consegue-se ver onde se encontra a ferramenta DEIprocessing no IDE. A Processing Foundation lançou uma nova versão, a 4.0 (a versão beta 2, a mais recente até à data, foi lançada a 5 de outubro de 2021 [70]), e que durante quase todo o tempo decorrido deste estágio se trabalhou sobre a versão 3.5.4. Ainda assim, como funcionou praticamente como uma ponte a nossa ferramenta desenvolvida continua funcional nesta versão mais recente (testada na versão 3.5.4 e na versão 4.0, apenas em Windows).

A criação desta ferramenta do IDE Processing exigiu uma grande pesquisa e aprofundamento de como estas ferramentas funcionavam [8, 9, 10, 11, 12, 13].

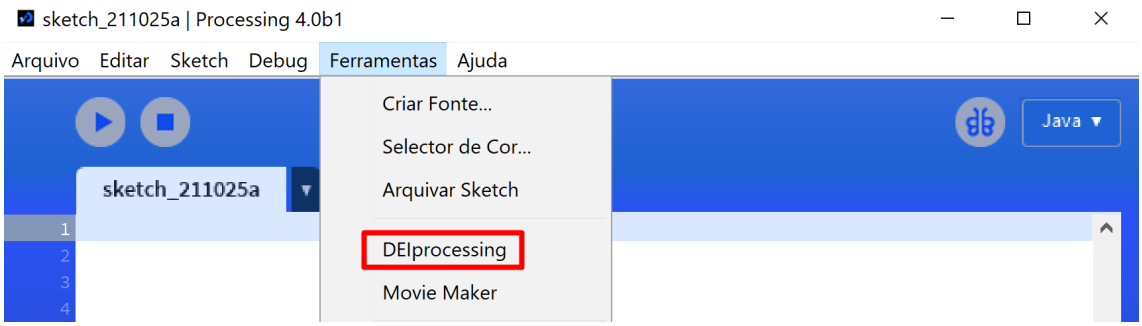


Figura 4.5: A ferramenta DEIprocessing disponível no IDE Processing

Esta página foi propositadamente deixada em branco.

Capítulo 5

Descrição da ferramenta

5.1 Apresentação da ferramenta DEIprocessing

A ferramenta desenvolvida, designada por DEIprocessing, encontra-se disponível para auxiliar a aprendizagem na linguagem de programação Processing.

A ferramenta DEIprocessing encontra-se disponível para download, no GitHub [71] da mesma, para todos aqueles que a quiserem utilizar e explorar, bem como uma explicação de como a manusear.

O download pode ser feito de duas formas: uma que inclui já o IDE Processing (com a versão 4.0) com a ferramenta DEIprocessing pronta a usar; ou então de forma faseada no Processing, que a pessoa já possui. Estes passos estão todos explicados na página de GitHub da ferramenta, como podemos ver na Figura 5.1.

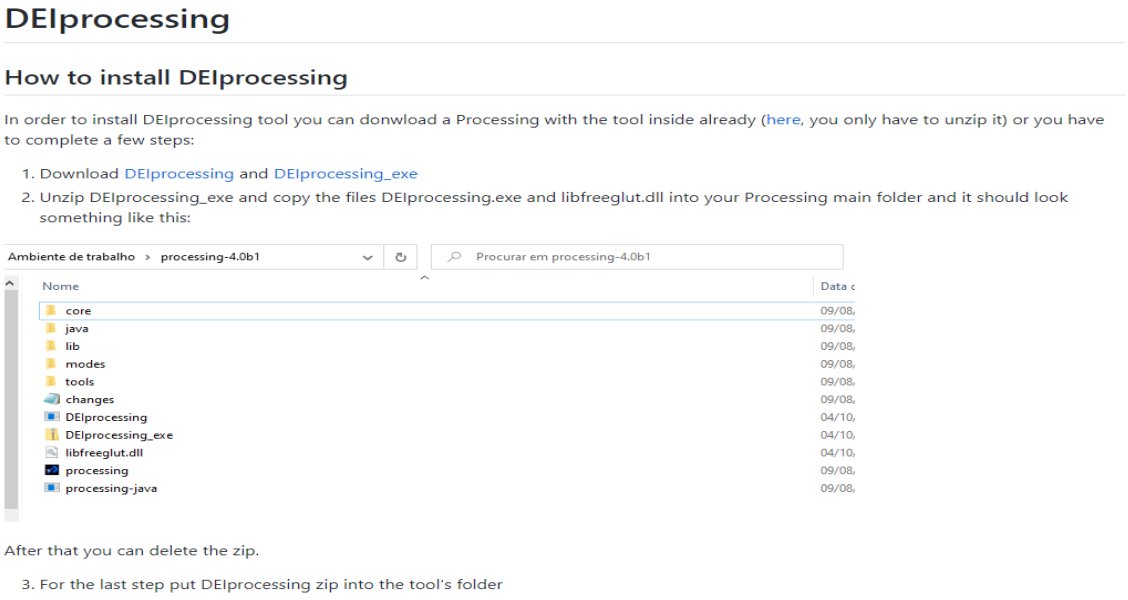


Figura 5.1: Primeiros passos da instalação da DEIprocessing (GitHub)

5.2 Interface da ferramenta DEIprocessing

O conteúdo da presente secção é referente à descrição da ferramenta criada de forma mais detalhada e como esta foi pensada e organizada (pode-se ver no apêndice B, 7.1, a forma como esta foi inicialmente idealizada). A ferramenta, à qual se deu o nome de DEIprocessing, está dividida em três partes, ou seja, três *viewports*. O principal (display1), o mais à esquerda da Figura 5.2, onde é representado o *output*, o do meio (display2), que é o que contém as variáveis (as do sistema e as definidas), o botão "NEXT" e ainda realça as mudanças consoante a linha em execução. O *viewport* mais à direita (display3) é onde se encontra o código executado e se realça a linha em execução.

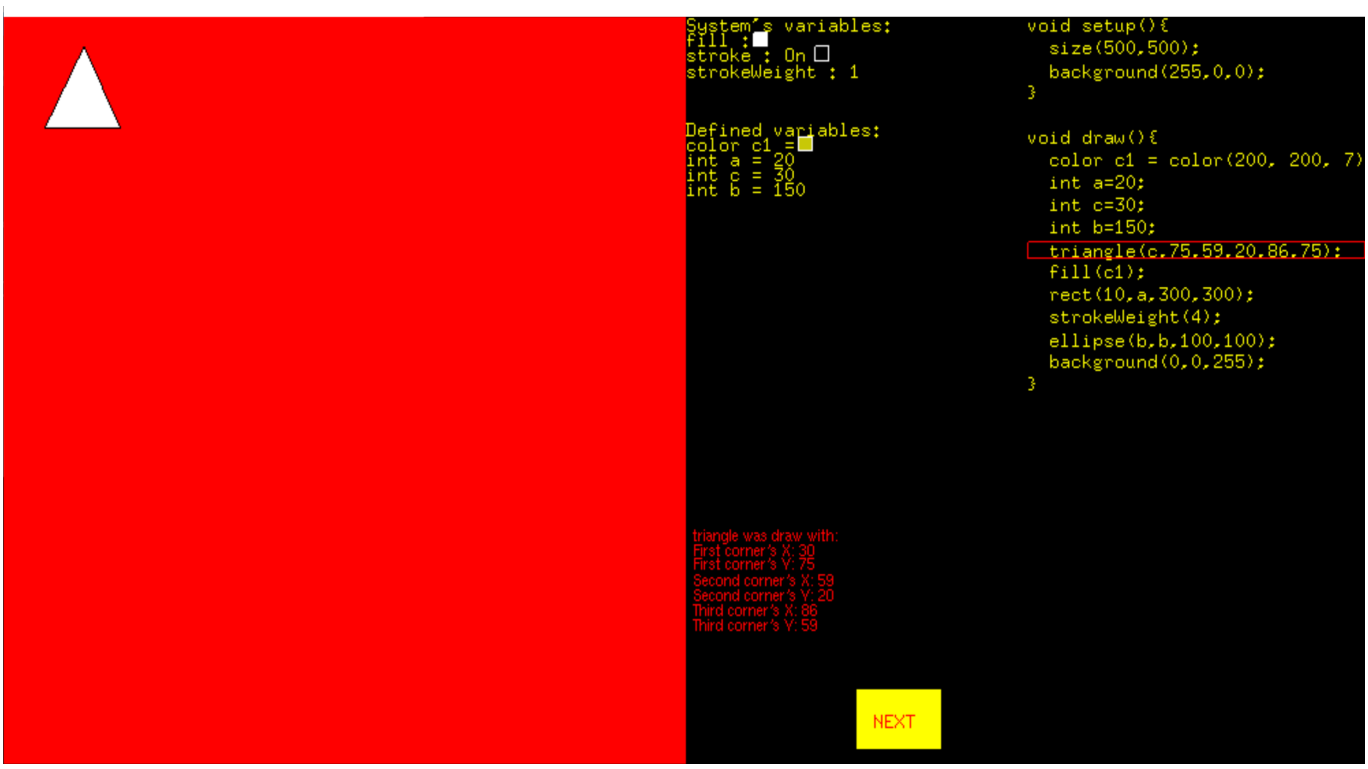


Figura 5.2: Interface da ferramenta DEIprocessing

A interface da ferramenta tem a intenção de ser atrativa e intuitiva, de forma a cumprir o objetivo de ajudar e facilitar o utilizador a compreender os programas simples escritos em Processing.

5.2.1 Display1

Este *viewport* é onde vemos, de forma interativa, o programa em execução e onde é possível ver as linhas de código a serem desenhas por *frames* controlados pelo utilizador.

Apesar de ser considerado o *display* principal, é aquele que aparenta ser mais claro e de fácil entendimento. O objetivo é a semelhante aparência com a do IDE Processing, porém de uma forma faseada e que também seja visualmente apelativa.

Todos os *displays* estão interligados e executam em simultâneo consoante aquilo que é pretendido.

5.2.2 Display2

O segundo display, o que se encontra no meio, é o que contém mais informação e o que interage diretamente com o utilizador.

Começando pelas variáveis, estas estão dividas da seguinte forma: variáveis do sistema e variáveis definidas. As primeiras estão sempre presentes independentemente do programa que está a correr, pois como o próprio nome indica são variáveis do sistema, inicialmente predefinidas pelo sistema. Em Processing existem mais do que apenas aquelas três mas, por uma questão de recursos limitados, ficaram apenas estas definidas na ferramenta DEI-processing. O outro tipo de variáveis é definido pelo utilizador no código.

Existe ainda aquilo que chamamos de *highlights* da linha do código, como podemos ver na Figura 5.2. Em função da linha de código que está a ser executada, os *highlights* mudam para evidenciar aquilo que está a acontecer no programa e em alguns casos, os parâmetros utilizados.

Por fim, existe um botão "NEXT", que é a zona que o utilizador deve clicar (com o botão direito do rato) quando pretende avançar para a próxima linha a executar no código. Carregar fora deste quadrado e/ou com o botão esquerdo não fará com que a ferramenta avance.

5.2.3 Display3

Do lado direito da Figura 5.2 encontra-se o último *viewport*, este existe para que seja possível proporcionar uma melhor experiência ao utilizador, uma vez que consiste em ter o código por ele escrito, na íntegra, representado mas com a particularidade da linha que está em execução ser assinalada a vermelho. Assim permite a quem usa a ferramenta consiga acompanhar que linha está a ser executada, o que esta faz (no display2) e o que mudou visualmente (no display1).

Esta página foi propositadamente deixada em branco.

Capítulo 6

Testes

Este capitulo descreve os testes relevantes realizados ao longo desta projeto. Dois tipos principais de testes foram feitos, os testes realizado por mim na ferramenta (DEIprocessing) e os testes efetuados a terceiros, neste caso aos alunos do primeiro ano de Licenciatura de Design e Multimédia (LDM) inscritos à unidade curricular de Introdução à Programação e Resolução de Problemas (IPRP).

Mais informação sobre os testes, bem como, detalhes sobre o procedimento desenvolvido está explanado nas próximas secções.

6.1 Testes iniciais

Consideram-se testes iniciais aqueles que foram feitos pela própria pessoa que desenvolveu a ferramenta/aplicação. Estes testes tiveram como objetivo de averiguar se certas partes da ferramenta em específico realizam o desempenho esperado.

6.1.1 Método MoSCoW

A priorização de funções e/ou tarefas, por vezes, pode ser desafiante e num projeto como este, com tempo limitado, era importante definirmos algumas prioridades, assim como limites. A implementação de novas funcionalidades levam sempre algum tempo e, como tal, o uso do método MoSCoW [72] foi importante para a organização e desenvolvimento da ferramenta.

Este método divide-se em 4 categorias[72]:

- Must have - Estes são considerados os requisitos mínimos, pelo que sem que estes estejam cumpridos assume-se que o projeto falhou e o produtor não está apto para ser usado. Quando considerados como "Must have", significa que são indispensáveis para um produto funcional.
- Should have - Nesta categoria encontram-se os requisitos que não são considerados essenciais, porém têm uma prioridade alta e existindo uma necessidade forte que sejam incluídos. Estes requisitos aumentam o valor do produto.
- Could have - Os "Could have" são requisitos que se devem considerar havendo tempo a mais, quando os "Must have" e os "Should have" já se encontram finalizados. Caso

não sejam alcançados, isto não significa um efeito negativo no produto.

- Won't have - Por fim, estes últimos consistem em desejos para o futuro que muitas vezes são impossíveis de realizar ou exigem muito tempo (ou dinheiro).

6.1.2 Testes realizados

Os ensaios feitos nesta categoria de testes encontram-se descritos nas Tabelas 6.1-6.2-6.3, na qual se usou o método MoSCoW para uma melhor perceção da importância daquilo que está a ser testado. Para além disto, a tabela contém o "ID" que permite a identificação única do teste em causa, uma breve descrição daquilo que se pretende com o teste, o código do programa em Processing onde foi testado e, por fim, o resultado na coluna designada de "Estado", que só pode ser "Funcional" ou "Não Funcional" consoante o resultado foi o esperado ou o não esperado, respetivamente.

Tabela 6.1: Tabela referente aos testes iniciais realizados na ferramenta DEIprocessing (parte 1)

ID	MoSCoW	Descrição	Código do programa em Processing	Estado

T1	Must have	Testar a função rect() da linguagem de programação Processing na ferramenta DEIprocessing	`void setup(){`
			`  size(500,500);`
			`  background(0,200,0);`
			`}`
			`void draw(){`
			`  int a=200;`
			`  rect(30,100,100,20);`
			`  rect(a,a,100,a);`
			`}`
				Funcional

T2	Must have	Testar a função triangle() da linguagem de programação Processing na ferramenta DEIprocessing	`void setup(){`
			`  size(500,500);`
			`  background(0,200,0);`
			`}`
			`void draw(){`
			`  int a=200;`
			`  triangle(a,75,59,20,86,75);`
			`  triangle(20,75,59,20,86,75);`
			`}`
				Funcional

T3	Must have	Testar a função ellipse() da linguagem de programação Processing na ferramenta DEIprocessing	`void setup(){`
			`  size(500,500);`
			`  background(0,200,0);`
			`}`
			`void draw(){`
			`  int a=200;`
			`  ellipse(a,a,100,100);`
			`  ellipse(50,50,100,100);`
			`}`
				Funcional

T4	Must have	Testar a função fill() da linguagem de programação Processing na ferramenta DEIprocessing	`void setup(){`
			`  size(500,500);`
			`  background(0,200,0);`
			`}`
			`void draw(){`
			`  int a=200;`
			`  color c1 = color(200, 200, 7);`
			`  fill(200,100,20);`
			`  ellipse(a,a,100,100);`
			`  fill(50);`
			`  ellipse(50,50,100,100);`
			`  fill(c1);`
			`  ellipse(100,100,100,100);`
			`}`
				Funcional

Tabela 6.2: Tabela referente aos testes iniciais realizados na ferramenta DEIprocessing (parte 2)

ID	MoSCoW	Descrição	Código do programa em Processing	Estado

T5	Must have	Testar a função background() da linguagem de programação Processing na ferramenta DEIprocessing	`void setup(){`
			`  size(500,500);`
			`  background(0,200,0);`
			`}`
			`void draw(){`
			`  color c1 = color(200, 200, 7);`
			`  background(255,0,0);`
			`  background(c1);`
			`  background(255,a,a);`
			`}`
				Funcional

T6	Should have	Testar as funções stroke() e noStroke() da linguagem de programação Processing na ferramenta DEIprocessing	`void setup(){`
			`  size(500,500);`
			`  background(0,200,0);`
			`}`
			`void draw(){`
			`  int a=200;`
			`  noStroke();`
			`  ellipse(a,a,100,100);`
			`  stroke(100);`
			`  ellipse(100,100,100,100);`
			`}`
				Funcional

T7	Should have	Testar a função strokeWeight() da linguagem de programação Processing na ferramenta DEIprocessing	`void setup(){`
			`  size(500,500);`
			`  background(0,200,0);`
			`}`
			`void draw(){`
			`  int a=200;`
			`  ellipse(a,a,100,100);`
			`  strokeWeight(3);`
			`  ellipse(100,100,100,100);`
			`}`
				Funcional

T8	Should have	Testar a estrutura de decisão If/Else da linguagem de programação Processing na ferramenta DEIprocessing	`void setup(){`
			`  size(500,500);`
			`  background(255,0,0);`
			`}`
			`void draw(){`
			`  color c2 = color(200, 0, 7);`
			`  int c=10;`
			`  if(c!=10){`
			`  fill(c2);`
			`  triangle(c,75,59,20,86,75);`
			`  }`
			`  else{`
			`  triangle(c,75,59,20,86,75);`
			`  rect(10,100,30,30);`
			`  }`
			`  fill(10);`
			`}`
				Funcional

T9	Could have	Testar a estrutura de iteração For da linguagem de programação Processing na ferramenta DEIprocessing	`void setup(){`
			`  size(500,500);`
			`  background(255,0,0);`
			`}`
			`void draw(){`
			`  color c1 = color(200, 200, 7);`
			`  color c2 = color(200, 0, 7);`
			`  int a=20;`
			`  for(int i=1;i<5;i=i+1){`
			`  ellipse(i,i,30,30);`
			`  ellipse(30,30,a,a);`
			`  fill(c2);`
			`  ellipse(a*i,a*i,100,100);`
			`  }`
			`  fill(c1);`
			`}`
				Funcional

Tabela 6.3: Tabela referente aos testes iniciais realizados na ferramenta DEIprocessing (parte 3)

ID	MoSCoW	Descrição	Código do programa em Processing	Estado
T10	Could have	Testar a função line() da linguagem de programação Processing na ferramenta DEIprocessing	Não implementada	Não Funcional
T11	Could have	Testar a função keyPressed() da linguagem de programação Processing na ferramenta DEIprocessing	Não implementada	Não Funcional
T12	Won't have	Testar a função mousePressed() da linguagem de programação Processing na ferramenta DEIprocessing	Não implementada	Não Funcional

Como se pode ver nas Tabelas 6.1-6.2-6.3, creio que se pode fazer um balanço positivo daquela que é a ferramenta DEIprocessing e considerar que o projeto foi bem sucedido, uma vez que todas as funções/estruturas testadas que tinha um nível de prioridade considerado alto ("Must have" e "Should have") demonstraram-se funcionais nos testes realizados, assim como a estrutura de iteração For que se encontrava no nível do "Could have".

6.2 Testes de usabilidade

Tendo em conta o tema desta dissertação associado à forte componente pedagógica nela presente, os testes de usabilidade junto dos estudantes são uma parte daquilo que se considera importante e que dá valor ao projeto. Como tal, foram realizados testes com alunos que frequentavam a cadeira de IPRP. Nesta secção iremos abordar o processo, os resultados e algumas conclusões mediante a apreciação recebida.

6.2.1 Processo

Os testes foram realizados de forma presencial, no Departamento de Engenharia Informática (DEI) da Faculdade de Ciências e Tecnologias da Universidade de Coimbra (FCTUC), em duas turmas, como já foi dito, de IPRP do primeiro ano da LDM, uma vez que ambas estão no início da aprendizagem da linguagem de programação Processing.

Para que fosse possível testar a ferramenta inicialmente, foi facultado o link do GitHub da ferramenta [71] aos alunos para fazer o seu download. Posteriormente, uma vez que já tinham a ferramenta instalada no IDE do Processing, foram sugeridos dois exemplos de código aos alunos, disponíveis na Figura 6.1, e foram encorajados a interagir com a ferramenta DEIprocessing.

```
void setup(){
  size(500,500);
  background(255,0,0);
}

void draw(){
  color c1 = color(200, 200, 7);
  color c2 = color(200, 0, 7);
  int a=20;
  int b=150;
  triangle(10,75,59,20,86,75);
  fill(c2);
  rect(10,a,300,300);
  strokeWeight(4);
  fill(c1);
  ellipse(b,b,100,100);
  background(0,0,255);
}
```

```
int px=30;
int r=30;
int py=250;

void setup(){
  size(500,500);
  fill(255,0,0);
}

void draw(){
  int passo=15;
  background(0);
  ellipse(px,py,2*r,2*r);
  if(px+passo<=500-r){
    px=px+passo;
    fill(255,0,0);
  }
}
```

Figura 6.1: Exemplos de códigos em Processing fornecidos aos alunos

Posteriormente foi pedido que os alunos respondessem ao questionário de *feedback* [73].

6.2.2 Resultados

Em relação aos resultados, obtivemos vinte sete respostas ao formulário, das nove perguntas, oito eram de carácter obrigatório, sete de escolha múltipla e duas de reposta aberta. Seguem-se as questões, assim como o parecer obtido.

De referir que as seis primeiras perguntas têm intencionalmente seis opções nas escalas respetivas para que o utilizador não tivesse a tendência de escolher a opção do meio e fosse obrigado a escolher entre "melhor" ou "pior", pois isso poderia levar a resultados inconclusivos.

Na 1ª pergunta, que era de escolha múltipla, os utilizadores tinham de responder numa escala de 1-6, onde 1 era "Péssima" e 6 "Excelente", à seguinte pergunta na Figura 6.2:

- Como classifica a ideia da criação da ferramenta DEIprocessing?

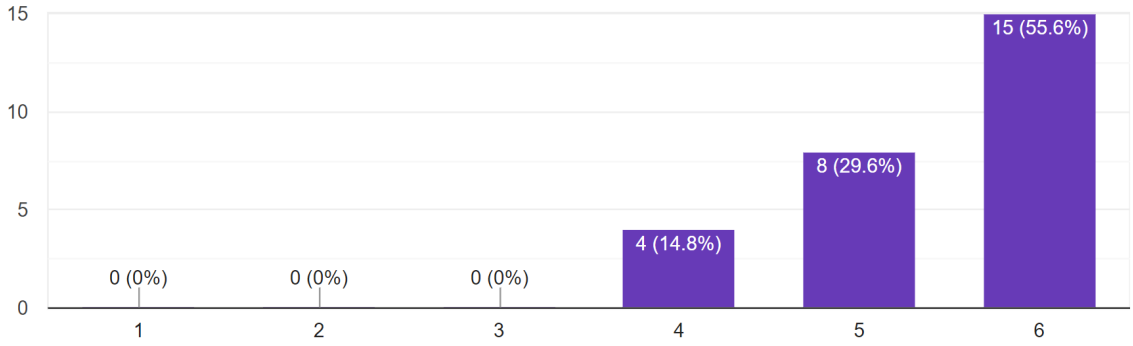


Figura 6.2: Resultados a 1ª pergunta

Na 2ª pergunta segue-se a mesma lógica da 1ª pergunta, mas para esta na Figura 6.3:

- Como classifica a interface da ferramenta DEIprocessing?

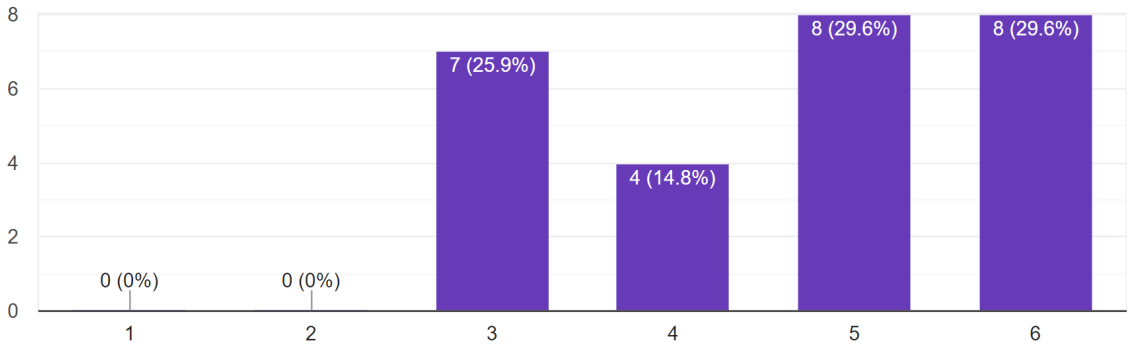


Figura 6.3: Resultados a 2ª pergunta

Na 3ª pergunta continua-se no mesmo registo das anteriores, na Figura 6.4:

- Como classifica o facto das linhas em execução, assim como as alterações feitas por essas linhas, serem realçadas/destacadas na ferramenta DEIprocessing?

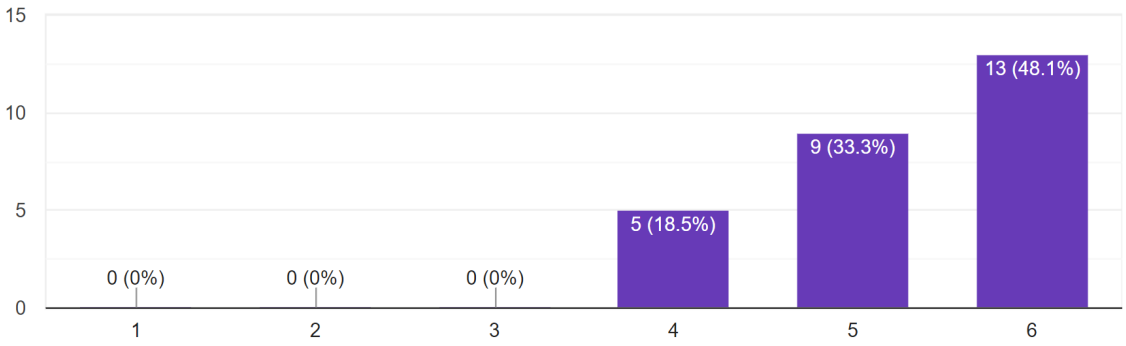


Figura 6.4: Resultados a 3ª pergunta

Na 4ª pergunta, mantém-se a escolha múltipla, bem como a escala de 1-6 porém agora o 1 significa "Nada" e o 6 "Bastante", com esta questão na Figura 6.5:

- O quanto útil acha a ferramenta DEIprocessing é na compreensão de programas simples escritos em Processing?

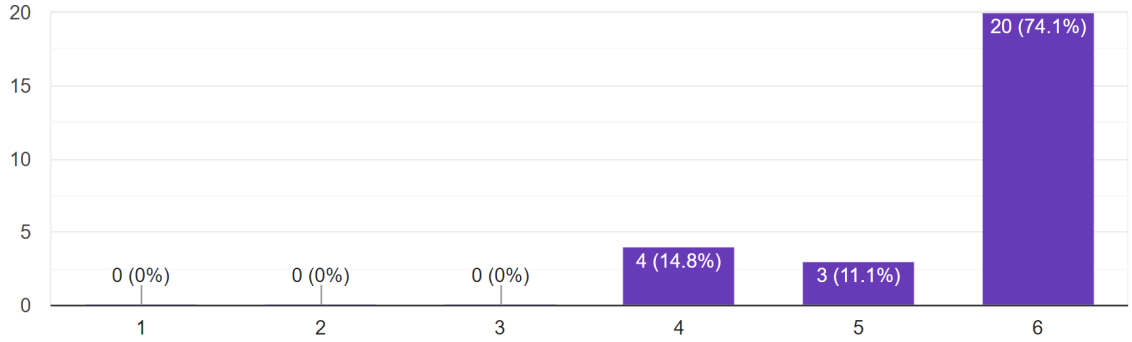


Figura 6.5: Resultados a 4ª pergunta

Na 5ª pergunta temos as mesmas características da anterior, na Figura 6.6:

- Numa escala de 1 a 6, quanto acha que a ferramenta DEIprocessing ajudaria na aprendizagem da linguagem Processing?

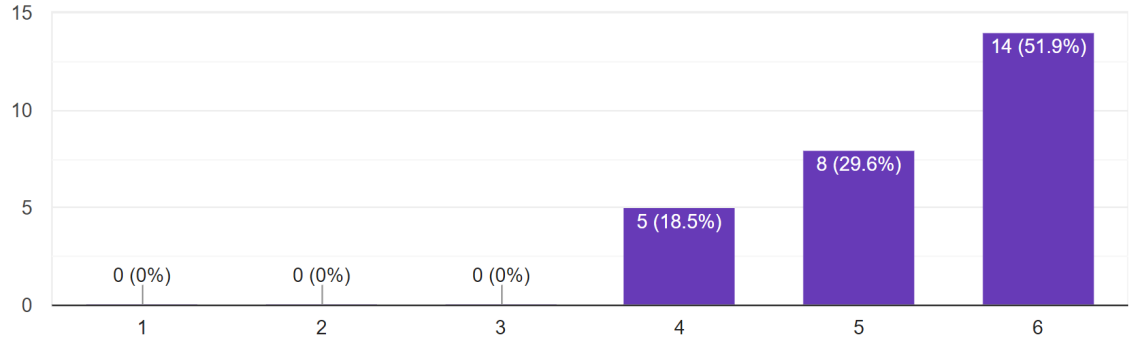


Figura 6.6: Resultados a 5ª pergunta

Na 6ª pergunta ainda temos as escolhas múltiplas, mas agora com apenas três opções Sim, Talvez e Não, como na Figura 6.7:

- Utilizaria a ferramenta DEIprocessing?

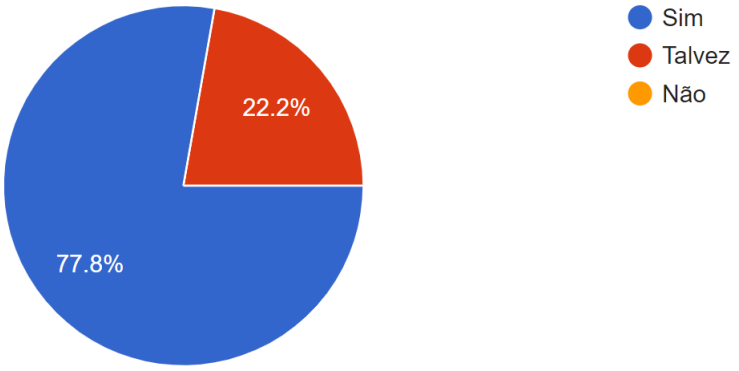


Figura 6.7: Resultados a 6ª pergunta

Na 7ª pergunta temos as mesmas opções que na 6ª, mas para esta presente na Figura 6.8:

- Gostaria que a ferramenta DEIprocessing continuasse a ser desenvolvida?

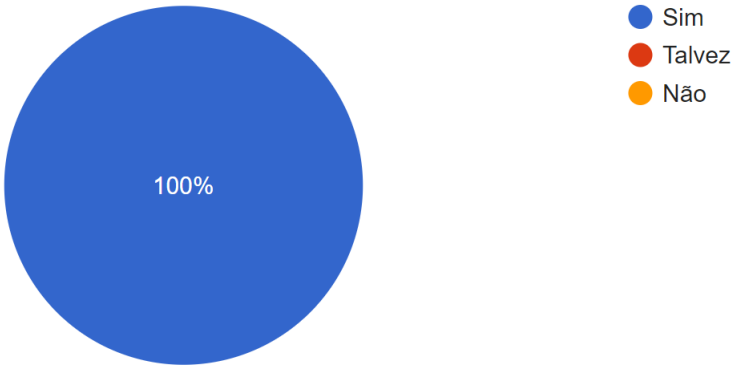


Figura 6.8: Resultados a 7ª pergunta

A 8ª pergunta é de resposta aberta para o seguinte:

- Indica o melhor e o pior aspeto da ferramenta DEIprocessing:

Os resultados da 8ª questão foram agrupados porque como era uma pergunta de resposta aberta, muitas das respostas significam o mesmo, apenas escritas com diferentes palavras. Como tal, temos:

- Melhor aspeto: Utilidade da ferramenta (16 pessoas), Simplicidade da ferramenta (4 pessoas), Fácil compreensão da ferramenta (2 pessoas) e Potencial da ferramenta (1 pessoa).
- Pior aspeto: A interface da ferramenta (11 pessoas), Não entendimento da ferramenta (2 pessoas), O botão "NEXT" da ferramenta só funcionar com o lado direito do rato (2 pessoas), Ferramenta só funcionar com programas simples (2 pessoas) e Ferramenta só andar para a frente (1 pessoa).

O facto da soma de cada tópico (melhor aspeto e pior) não dar as 27 respostas, deve-se ao facto de algumas respostas serem inválidas e por isso não as contabilizamos nesta pergunta.

A 9ª pergunta é novamente de resposta aberta e a única de carácter não obrigatório:

- Comentários adicionais:

Os únicos dois resultados que consideramos válidos nesta última questão foram:

- Espetacular.
- Uma *box* a explicar o funcionamento seria agradável.

A Tabela 6.4 mostra as médias, as medianas e as modas dos resultados recolhidos. Na pergunta 8ª e 9ª não é possível fazer este tipo de estudo.

Tabela 6.4: Tabela referente à média, à mediana e à moda das respostas obtidas

	Média	Mediana	Moda
1ª pergunta	5,407	6	6
2ª pergunta	4,630	5	5 e 6
3ª pergunta	5,296	5	6
4ª pergunta	5,593	6	6
5ª pergunta	5,333	6	6
6ª pergunta	-	"Sim"	"Sim"
7ª pergunta	-	"Sim"	"Sim"

6.2.3 Conclusões

O parecer obtido, na generalidade, demonstrou-se bastante positivo. Mais de 50% dos alunos consideraram "Excelente" a ideia desenvolvida nesta dissertação, enquanto os restantes também se mostram agradados, não havendo nenhuma classificação abaixo de 4 (numa escala de 1-6, como já referido anteriormente). Estes dados fazem com que uma das características pretendidas na ferramenta DEIprocessing tenha sido atingida, que é ser apelativa, os alunos manifestaram entusiasmo pelo desenvolvimento da mesma. Podemos confirmar estas afirmações já que nas perguntas seis e sete, quando questionamos se utilizariam a ferramenta DEIprocessing e se gostariam que continuasse a ser desenvolvida, respetivamente, todas as 27 respostas são "Sim" à exceção de 6 pessoas que responderam "Talvez" na 6ª pergunta.

Quando questionados sobre a interface da ferramenta DEIprocessing, percebe-se, juntamente com a pergunta sobre o melhor/pior aspeto, que é o ponto menos positivo da ferramenta. Não deixa de ter bons resultados, no entanto a interface não é o principal objetivo deste projeto e para que se atingisse um nível melhor seria necessário a colaboração de um designer. Por outro lado, nas perguntas sobre a ajuda na compreensão de programas simples e na aprendizagem da linguagem Processing existe uma concordância nas respostas que demonstra que estas características foram atingidas de forma bastante satisfatória. Ainda sobre este assunto, na pergunta de resposta aberta sobre o melhor aspeto, quase todas as respostas remetem para a utilidade da ferramenta nos pontos referidos anteriormente (compreensão e aprendizagem).

Falta apenas falar de uma pergunta de escolha múltipla que aborda duas funcionalidade da DEIprocessing: o realce da linha de código em execução e o realce das alterações feitas por essa linha de código. A opinião geral dos alunos foi, novamente, muito boa e consegue-se deduzir que a grande maioria percebeu como estas funcionalidades podem ajudar na aprendizagem da linguagem de programação Processing.

Por fim, foi possível apreender o interesse geral dos alunos com a ferramenta DEIprocessing e que o seu propósito foi alcançado mesmo que existam, como sempre existiram, aspetos que podem ser alterados e/ou mudados, assim como mencionado em algumas respostas e caso o projeto continue a ser desenvolvido após o término da dissertação pretende-se que sejam abordados.

Esta página foi propositadamente deixada em branco.

Capítulo 7

Conclusão

Neste documento fala-se da visualização interativa de programas escritos em Processing. Como tal, foi feito um estudo sobre o estado da arte do Processing que permitiu estabelecer algumas ideias e objetivos, e que incluiu tutoriais, em particular o "Hello Processing!" e o "Getting Started", e ferramentas estudadas, como por exemplo o *turtle* e o visualgo. Nesta pesquisa foi ponderada a criação de uma biblioteca em Processing, que acabou por não ter efeito. Ainda no estado da arte encontram-se os aspetos pedagógicos estudados nesta dissertação. Toda esta investigação contribuiu para o avanço do projeto e da especificação das funcionalidades implementadas no segundo semestre.

O presente relatório contém uma parte introdutória sobre o Processing e a sua história, assim como, o planeamento dos semestres e as alterações feitas do primeiro para o segundo semestre, com auxílio a diagramas de Gantt. Conta ainda com um capítulo dedicado ao desenvolvimento da ferramenta fruto desta dissertação e outro capítulo onde esta se descreve. Por fim são abordados os testes realizados.

O primeiro semestre foi divido em cinco grandes partes: familiarização com o Processing, através de tutorias e alguns exercícios, estudo do estado da Arte, como já referido acima, e a adaptação com as tecnologias a usar, nomeadamente as necessárias para a criação de um tradutor, desenvolvimento do tradutor, análise lexical e análise sintática (ficheiros lex e bison), e ainda a iniciação de uma Abstract Syntax Tree (AST), e o relatório intermédio.

O segundo semestre é uma continuação do trabalho realizado no primeiro, no qual se dá continuação à tradução orientada pela sintaxe (AST), seguida da análise semântica e em simultâneo procedendo as alterações necessárias na análise lexical e análise sintática. Assim como no primeiro semestre foi necessário a aprendizagem com novas tecnologias, também neste semestre foi necessário fazê-lo em relação ao OpenGL, que deu origem às seguintes etapas, desenho interativo, visualização da correlação código-input e *highlights* da linha de código. Ainda durante este semestre foi criada a ferramenta no IDE Processing e foram realizados os testes necessários uma vez que era funcional. Por fim, procedeu-se aos refinamentos da ferramenta DEIprocessing e à redução a escrito do relatório final da dissertação.

Este estudo pode ser considerado um passo inicial no desenvolvimento do auxilio da aprendizagem de programação em Processing, através de ferramentas, pois foi possível durante esta dissertação demonstrar o potencial e o sucesso que este tipo de projetos podem ter junto dos alunos, conforme se verificou pelo apreciação obtida nos testes feitos.

Em suma, acredita-se que a investigação feita e as informações recolhidas permitiram a escolha de uma abordagem correta e a criação de um tradutor, bem como, posteriormente,

de uma ferramenta que se assume que conseguiu atingir os objetivos desta dissertação de desenvolver uma ferramenta que fosse possível construir e simular programas simples em Processing, conseguindo simulá-los instrução a instrução, de modo a que os utilizadores possam, de forma mais facilitada e interativa, ver como as diferentes instruções são executadas e produzem determinados resultados. Não obstante, existe espaço para melhorias, apresentando este trabalho uma base sólida para o projeto puder continuar a ser desenvolvido.

7.1 Trabalho Futuro

Como já dito anteriormente esta dissertação, após 14 meses de trabalho, demonstrou que projeto desenvolvido está muito longe de ser dado como terminado, atenta a sua complexidade, assim como ao facto de existir sempre a possibilidade de melhorar e acrescentar novas funcionalidades.

Quando se fala de melhorias pode-se referir alguns temas mencionados nas respostas obtidas dos alunos, como por exemplo a interface, em colaboração com um designer podia beneficiar bastante, como já se tinha referido. O próprio tradutor poderia continuar a ser desenvolvido e como tal, conseguir traduzir mais métodos da linguagem de programação Processing e, eventualmente, aceitar toda a linguagem em si e traduzi-la para OpenGL.

Nomeadamente quanto às funcionalidades futuras, assim como às melhorias, foram referidas no parecer dos alunos algumas ideias como a possibilidade de, para além de poder andar de linha em linha para a frente no código, poder andar-se também para trás. E também a possibilidade de acrescentar funcionalidades não ponderadas até à data.

A abordagem como este projeto foi desenvolvido dá o controlo total sobre aquilo que está a ser trabalho e, consequentemente tem-se uma liberdade de expansão e criatividade maior no futuro. Sucede porém que, que tal processo exige mais recursos, designadamente humanos e financeiros, e mais tempo.

Referências

[1] História do cisuc. https://www.cisuc.uc.pt/en/history.

[2] Overview do processing. https://processing.org/overview/. Antes de 09-08-2021.

[3] Teaching computing with processing, the bridge between high school and college (abstract only). https://dl.acm.org/doi/10.1145/2676723.2678286.

[4] Wikipédia do mit media lab. https://en.wikipedia.org/wiki/MIT_Media_Lab.

[5] Wikipédia do processing. https://en.wikipedia.org/wiki/Processing_(programming_language)#:~:text=Processing%20is%20an%20open%2Dsource,programming%20in%20a%20visual%20context.

[6] Rui Gustavo Crespo. *Processadores de linguagens*. IST - Instituto Superior Técnico (January 1, 2001), 2001.

[7] Overview das ferramentas do processing. https://github.com/processing/processing/wiki/Tool-Overview.

[8] O essencial das ferramentas do processing. https://github.com/processing/processing/wiki/Tool-Basics.

[9] Ferramentas do processing. https://processing.org/reference/tools/.

[10] Github da classe base das ferramentas do processing. https://github.com/processing/processing/blob/master/app/src/processing/app/Base.java.

[11] Github da classe preferences das ferramentas do processing. https://github.com/processing/processing/blob/master/app/src/processing/app/Preferences.java.

[12] Github da classe sketchcode das ferramentas do processing. https://github.com/processing/processing/blob/master/app/src/processing/app/SketchCode.java.

[13] Github da classe sketch das ferramentas do processing. https://github.com/processing/processing/blob/master/app/src/processing/app/Sketch.java.

[14] Análise de mercado. http://www.netmba.com/marketing/market/analysis/.

[15] Yizhou Qian and James Lehman. Students' misconceptions and other difficulties in introductory programming: A literature review. Technical report, 2017.

[16] Masura Rahmat, Shahrina Shahrani, Rodziah Latih, Noor Faezah Mohd Yatim, Noor Faridatul Ainun Zainal, and Rohizah Ab Rahman. Major problems in basic programming that influence student performance. Technical report, 2011.

[17] Tutorial happy coding. `https://happycoding.io/tutorials/processing/`.

[18] Tutorial fun programming. `https://funprogramming.org/`.

[19] Tutorial do processing. `https://processing.org/tutorials/`.

[20] Tutorial "hello processing!". `https://hello.processing.org/editor/`.

[21] Tutorial "getting started". `https://github.com/jaewhyun/GettingStarted`.

[22] About do hello processing. `https://hello.processing.org/about/`.

[23] Website processing "getting started". `https://processing.org/tutorials/gettingstarted/`.

[24] Tools do processing. `https://processing.org/reference/tools/`.

[25] Turtle. `https://docs.python.org/3/library/turtle.html#module-turtle`.

[26] Wikipédia do turtle. `https://en.wikipedia.org/wiki/Turtle_graphics`.

[27] Página do visualgo. `https://visualgo.net/en`.

[28] Exemplo do visualgo. `https://visualgo.net/en/sorting`.

[29] Interactive animations (anim). `http://anim.ide.sk/introduction.php`.

[30] Exemplo da interactive animations (anim). `http://anim.ide.sk/basic_algorithms.php`.

[31] Anim sorting cards. `http://anim.ide.sk/sortingcards.php`.

[32] Anim sorting algorithms 1. `http://anim.ide.sk/sorting_algorithms_1.php`.

[33] Anim sorting algorithms 2. `http://anim.ide.sk/sorting_algorithms_2.php`.

[34] Anim crossing river. `http://anim.ide.sk/crossing.php`.

[35] Casey Reas and Ben Fry. Processing: Programming for the media arts. *AI and Society*, 20, 2006.

[36] Adriana J. Berlanga and Francisco J. García. Authoring tools for adaptive learning designs in computer-based education. *ACM International Conference Proceeding Series*, 124, 2005.

[37] M. H. Brown. Perspectives on algorithm animation. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, -, 1988.

[38] T. A. Budd. An active learning approach to teaching the data structures course. *In Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, -, 2006.

[39] Linda Null and Julia Lobur. Mariesim: The marie computer simulator. *J. Educ. Resour. Comput.*, 3, 2003.

[40] Dino Schweitzer, Jeff Boleng, and Lauren Scharff. Interactive tools in the graphics classroom. *ITiCSE'11 - Proceedings of the 16th Annual Conference on Innovation and Technology in Computer Science*, -, 2011.

[41] Processing. `http://processing.org`.

[42] P5.js. http://p5js.org/.

[43] Casey Reas. The language of computers. *Creative Code, John Maeda and Red Burns (Eds.)*, 44, 2004.

[44] C. Reas and B. Fry. The processing handbook, 2007.

[45] Jingyi Li, Joel Brandt, Radomír Mech, Maneesh Agrawala, and Jennifer Jacobs. Supporting visual artists in programming through direct inspection and control of program execution. *Conference on Human Factors in Computing Systems - Proceedings*, -, 2020.

[46] Qurat ul Ain, Farah Shahid, Muhammad Aleem, Muhammad Arshad Islam, Muhammad Azhar Iqbal, and Muhammad Murtaza Yousaf. A review of technological tools in teaching and learning computer science. *Eurasia Journal of Mathematics, Science and Technology Education*, 15, 2019.

[47] R. C. Richey, K. H. Silber, and D. P. Ely. Reflections on the 2008 aect definitions of the field. *TechTrends*, 52, 2008.

[48] V. Marcy. Introdução ao vark. https://vark-learn.com/introduction-to-vark/.

[49] Thomas Suselo, Burkhard C. Wünsche, and Andrew Luxton-Reilly. Technologies and tools to support teaching and learning computer graphics: A literature review. *ACM International Conference Proceeding Series*, -, 2019.

[50] Ruth Torres Castillo. Computer games as learning tools: Teachers attitudes and behaviors. *CHI PLAY 2018 - Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*, -, 2018.

[51] Daniel Rees Lewis, Emily Harburg, Elizabeth Gerber, and Matthew Easterday. Building support tools to connect novice designers with professional coaches. *C and C 2015 - Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition*, -, 2015.

[52] Formulário de dificuldades na aprendizagem em processing. https://forms.gle/2M8PK5aMDQAmEBPY7.

[53] Tutorial sobre bibliotecas em processing parte 1. https://www.youtube.com/watch?v=pI2gvl9sdtE&ab_channel=TheCodingTrain.

[54] Tutorial sobre bibliotecas em processing parte 2. https://www.youtube.com/watch?v=U0TGZCEWn8g&ab_channel=TheCodingTrain.

[55] Discussão do processing 3. https://www.youtube.com/watch?v=LOcSnePMfb0&ab_channel=TheCodingTrain.

[56] Tutorial sobre a classe papplet. https://www.geeksforgeeks.org/how-to-create-a-papplet-project-in-eclipse-processing/.

[57] Github da classe papplet core. http://processing.github.io/processing-javadocs/core/.

[58] Github da classe papplet. https://github.com/processing/processing/blob/94144be998354ecf7a06d5301b7b4fab72df.

[59] Wikipédia do bell labs. https://en.wikipedia.org/wiki/Bell_Labs.

[60] Wikipédia do gnu. https://en.wikipedia.org/wiki/GNU.

[61] Wikipédia do lex. https://en.wikipedia.org/wiki/Lex_(software).

[62] Gnu bison. https://www.gnu.org/software/bison/.

[63] Wikipédia do bison. https://en.wikipedia.org/wiki/GNU_Bison.

[64] Wikipédia da ast. https://en.wikipedia.org/wiki/Abstract_syntax_tree.

[65] Wikipédia da análise semântica. https://pt.wikipedia.org/wiki/An%C3%A1lise_sem%C3%A2ntica.

[66] Wikipédia do opengl. https://en.wikipedia.org/wiki/OpenGL.

[67] Instalação do opengl e freeglut. https://medium.com/@bhargav.chippada/how-to-setup-opengl-on-mingw-w64-in-windows-10-64-bits-b77f350cea7e.

[68] Wikipédia do grupo khronos. https://en.wikipedia.org/wiki/Khronos_Group.

[69] Wikipédia do freeglut. https://en.wikipedia.org/wiki/FreeGLUT.

[70] Versões e downloads do processing. https://processing.org/download.

[71] Github da ferramenta deiprocessing. https://github.com/DuarteCarvalhoo/DEIprocessing.

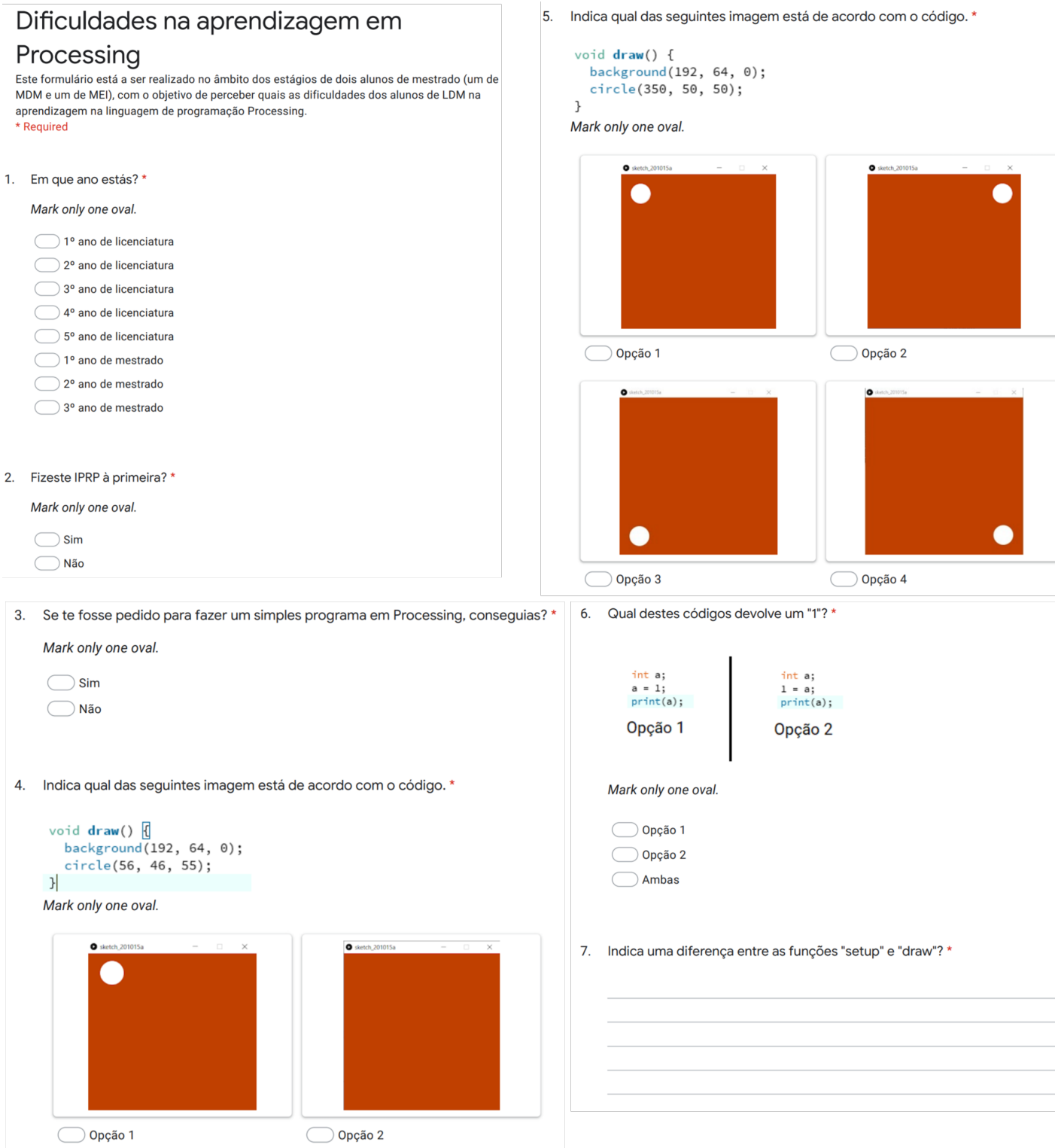
[72] Método moscow. https://www.toolshero.com/project-management/moscow-method/.

[73] Formulário de feedback da ferramenta deiprocessing. https://forms.gle/gFMaS2RG4aKyB7Tz6.

Appendices

Esta página foi propositadamente deixada em branco.

Apêndice A



Dificuldades na aprendizagem em Processing

Este formulário está a ser realizado no âmbito dos estágios de dois alunos de mestrado (um de MDM e um de MEI), com o objetivo de perceber quais as dificuldades dos alunos de LDM na aprendizagem na linguagem de programação Processing.

* Required

1. Em que ano estás? *

Mark only one oval.

- 1º ano de licenciatura
- 2º ano de licenciatura
- 3º ano de licenciatura
- 4º ano de licenciatura
- 5º ano de licenciatura
- 1º ano de mestrado
- 2º ano de mestrado
- 3º ano de mestrado

2. Fizeste IPRP à primeira? *

Mark only one oval.

- Sim
- Não

3. Se te fosse pedido para fazer um simples programa em Processing, conseguias? *

Mark only one oval.

- Sim
- Não

4. Indica qual das seguintes imagem está de acordo com o código. *

```
void draw() {
  background(192, 64, 0);
  circle(56, 46, 55);
}
```

Mark only one oval.

- Opção 1
- Opção 2

5. Indica qual das seguintes imagem está de acordo com o código. *

```
void draw() {
  background(192, 64, 0);
  circle(350, 50, 50);
}
```

Mark only one oval.

- Opção 1
- Opção 2
- Opção 3
- Opção 4

6. Qual destes códigos devolve um "1"? *

```
int a;
a = 1;
print(a);
```

Opção 1

```
int a;
1 = a;
print(a);
```

Opção 2

Mark only one oval.

- Opção 1
- Opção 2
- Ambas

7. Indica uma diferença entre as funções "setup" e "draw"? *

Figura 1: Questionário "Dificuldades na aprendizagem em Processing

Esta página foi propositadamente deixada em branco.

Apêndice B

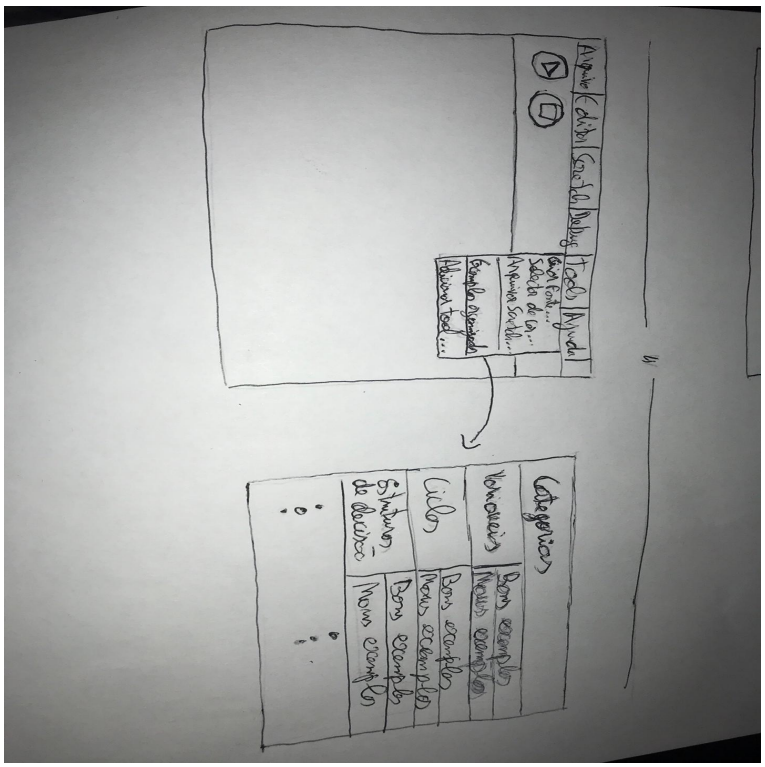


Figura 2: *Mockup* do IDE Processing e onde estaria a ferramenta DEIprocessing

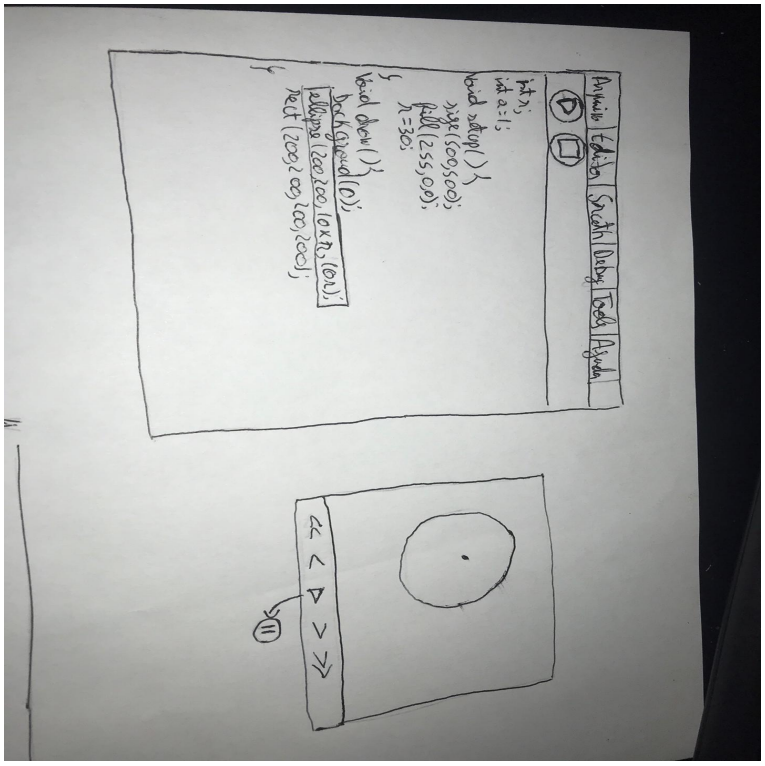


Figura 3: *Mockup* da interface da ferramenta DEIprocessing

Esta página foi propositadamente deixada em branco.

Apêndice C

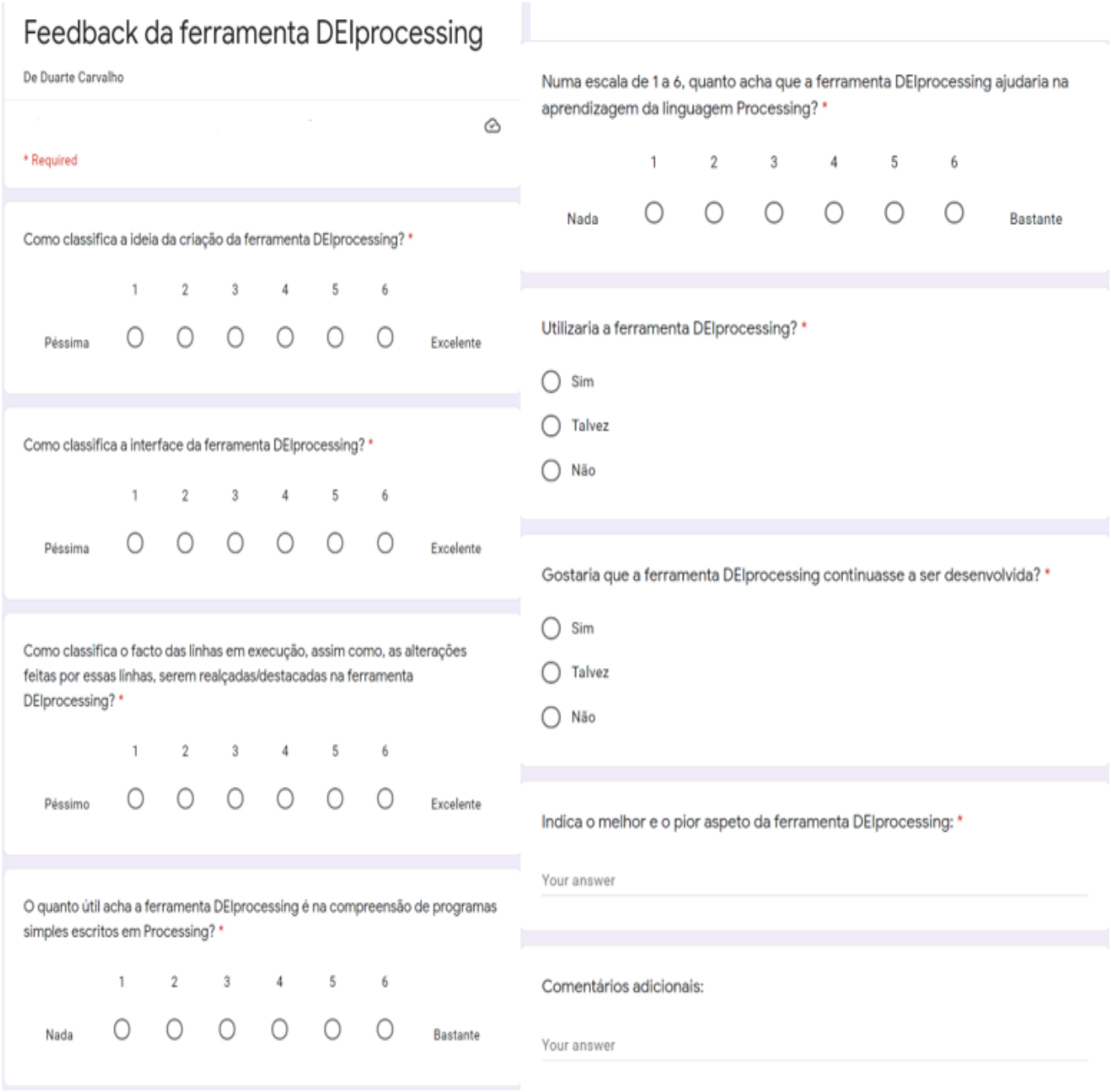


Figura 4: Questionário "*Feedback* da ferramenta DEIprocessing"