Uma ferramenta para o ensino de programação: concepção, desenvolvimento e avaliação

Luís Carlos Lacerda Durans

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 – 50732-970 – Recife – PE – Brasil

lcld@cin.ufpe.br

Abstract. *This article presents the development of a tool/plugin for Unity that allows teaching programming in an educational way, through a visual coding interface in blocks in real time. The tool facilitates the creation of games that teach programming, as well as enabling the learning of concepts and structures of programming in an interactive and playful way. The article describes the conception, development, and evaluation of the tool, which was tested with novice programming users and assessed according to their experience. The results showed that the tool was fun, didactic and effective, increasing the interest, motivation and retention of information of the users. The article concludes that the tool is an innovative and creative proposal for teaching programming in Unity, which can contribute to the training of programmers with skills and competencies to solve problems, create solutions and develop projects in different areas.*

Resumo. *Este artigo apresenta o desenvolvimento de uma ferramenta/plugin para Unity que permite o ensino de programação de forma educativa, por meio de uma interface de codificação visual em blocos em tempo de execução. A ferramenta facilita a criação de jogos que ensinam programação, além de possibilitar o aprendizado de conceitos e estruturas de programação de forma interativa e lúdica. O artigo descreve a concepção, o desenvolvimento e a avaliação da ferramenta, a qual foi testada com usuários iniciantes em programação e analisada conforme a experiência deles. Os resultados mostraram que a ferramenta foi divertida, didática e eficaz, aumentando o interesse, a motivação e a retenção de informação dos usuários. O artigo conclui que a ferramenta é uma proposta inovadora e criativa para o ensino de programação em conjunto com a Unity, que pode contribuir para a formação de programadores com habilidades e competências para resolver problemas, criar soluções e desenvolver projetos em diferentes áreas.*

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Durans, Luís Carlos Lacerda.
uma ferramenta para o ensino de programação: concepção, desenvolvimento e avaliação / Luís Carlos Lacerda Durans. - Recife, 2023.
15 p. : il., tab.

Orientador(a): Sílvio de Barros Melo
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Ciências da Computação - Bacharelado, 2023.

1. Educação. 2. Unity. 3. Programação. 4. Codificação Visual. I. Barros Melo, Sílvio de. (Orientação). II. Título.

000 CDD (22.ed.)

1. Introdução

A programação é uma habilidade cada vez mais importante e demandada na sociedade atual, que envolve o uso de linguagens e ferramentas para criar soluções e projetos em diferentes áreas. O ensino de programação, porém, enfrenta diversos desafios, como a complexidade, a abstração, a sintaxe e a lógica das linguagens de programação, que podem dificultar o aprendizado e desmotivar os estudantes. Além disso, o ensino de programação muitas vezes é feito de forma isolada e descontextualizada, sem relação com os interesses e as necessidades dos estudantes. Nesse sentido, é preciso buscar formas de tornar o ensino de programação mais atrativo, significativo e efetivo, que estimulem o interesse, a motivação e a retenção de informação dos estudantes. Uma das formas possíveis é o uso de jogos digitais, que são produtos de programação, mas também podem ser ferramentas de ensino e aprendizagem [Alves 2013] [Moreira 2016] [Maloney et al. 2010].

Este artigo apresenta o desenvolvimento de uma ferramenta/plugin para Unity que permite o ensino de programação de forma educativa, por meio de uma interface de codificação visual em blocos em tempo de execução. A Unity é uma plataforma de criação de jogos digitais, que oferece recursos gráficos, físicos, sonoros e de inteligência artificial, além de suportar diversas linguagens de programação, como C#, JavaScript e Boo [Unity Technologies 2020]. A ferramenta/plugin desenvolvida neste trabalho facilita a criação de jogos que ensinam programação, além de possibilitar o aprendizado de conceitos e estruturas de programação de forma interativa e lúdica. Ela se baseia na abordagem de codificação visual, que consiste em usar elementos gráficos, como blocos, ícones, cores e formas, para representar e manipular elementos de programação, como variáveis, operadores, funções e estruturas de controle [Berglund and Wiggberg 2018, Moreno-León et al. 2015]. A codificação visual permite que os estudantes se concentrem na lógica e na criatividade, sem se preocupar com a sintaxe e os erros das linguagens de programação [Sáez-López et al. 2016].

A relevância deste trabalho está em contribuir para a formação de programadores com habilidades e competências para resolver problemas, criar soluções e desenvolver projetos em diferentes áreas. A hipótese deste trabalho é que a ferramenta/plugin desenvolvida aumenta o interesse, a motivação e a retenção de informação dos estudantes, tornando o ensino de programação mais divertido, didático e eficaz. A metodologia utilizada neste trabalho foi a concepção, o desenvolvimento e a avaliação da ferramenta/plugin, que foi testada com usuários iniciantes em programação, para coletar suas opiniões/críticas acerca do método proposto. A estrutura deste trabalho: na seção 2, é feita uma revisão da literatura sobre o ensino de programação, a codificação visual e a Unity; na seção 3, é descrita a concepção, o desenvolvimento e a avaliação da ferramenta/plugin; na seção 4, são apresentados e discutidos os resultados obtidos; na seção 5, são feitas as considerações finais e as sugestões para trabalhos futuros.

2. Revisão de Literatura

Nesta seção, serão apresentados alguns conceitos e estudos relacionados ao ensino de programação, à codificação visual e à Unity, que servem de embasamento teórico para este trabalho.

2.1. Ensino de Programação

O ensino de programação é uma atividade que visa desenvolver habilidades e competências para criar soluções e projetos em diferentes áreas, utilizando linguagens e ferramentas computacionais. Além disso, o ensino de programação pode favorecer o desenvolvimento de habilidades socioemocionais, como a colaboração, a comunicação, a criatividade e a persistência [Kelleher and Pausch 2005, Resnick et al. 2009].

Uma das formas tradicionais de ensinar programação é o uso de linguagens textuais, que exigem que os estudantes escrevam códigos seguindo regras sintáticas e semânticas específicas. Essas linguagens podem ser classificadas em diferentes paradigmas, como imperativo, funcional, lógico e orientado a objetos, que refletem diferentes formas de pensar e resolver problemas [Van Roy and Haridi 2004]. O ensino de programação baseado em linguagens textuais pode apresentar algumas vantagens, como o desenvolvimento de habilidades de abstração, de atenção aos detalhes, de depuração e de documentação [McCracken et al. 2001]. No entanto, esse tipo de ensino também pode apresentar algumas desvantagens, como a dificuldade de compreender e memorizar a sintaxe, a frustração e a perda de tempo com erros de digitação, a falta de feedback imediato e de visualização dos resultados, e a pouca conexão com o mundo real e com os problemas relevantes [Gomes and Mendes 2007, Sorva et al. 2013].

2.2. Codificação Visual

Uma das formas alternativas de ensinar programação é o uso da codificação visual, que consiste em usar elementos gráficos, como blocos, ícones, cores e formas, para representar e manipular elementos de programação, como variáveis, operadores, funções e estruturas de controle [Berglund and Wiggberg 2018, Moreno-León et al. 2015]. A codificação visual permite que os estudantes se concentrem na lógica e na criatividade, sem se preocupar com a sintaxe e os erros das linguagens de programação. A codificação visual também facilita a compreensão e a comunicação dos conceitos e processos de programação, por meio de uma linguagem mais próxima da humana e de uma disposição mais intuitiva e organizada [Bau et al. 2017, Maloney et al. 2010]. Alguns exemplos de ferramentas de codificação visual são o Scratch, o Blockly e o App Inventor, que permitem criar programas e aplicativos de forma interativa e lúdica, usando uma interface de arrastar e soltar blocos [Resnick et al. 2009].

O ensino de programação baseado em codificação visual pode apresentar algumas vantagens, como o aumento do interesse, da motivação e da autoconfiança dos estudantes, a facilitação do aprendizado de conceitos e estruturas de programação, a promoção da criatividade, da colaboração e do pensamento computacional, e a possibilidade de integração com outras áreas do conhecimento e com problemas autênticos. No entanto, esse tipo de ensino também pode apresentar algumas desvantagens, como a limitação das funcionalidades e da expressividade das ferramentas, a dificuldade de transição para linguagens textuais, a falta de padronização

e de documentação, e a pouca avaliação e validação dos resultados [Berglund and Wiggberg 2018] [Weintrop and Wilensky 2015].

2.3. Unity

A Unity é uma plataforma de criação de jogos digitais, que oferece recursos gráficos, físicos, sonoros e de inteligência artificial, além de suportar diversas linguagens de programação, como C#, JavaScript e Boo. A Unity é uma ferramenta poderosa e flexível, que permite desenvolver jogos e aplicativos para diferentes plataformas, como desktop, mobile, web, console e realidade virtual e aumentada. A Unity também possui uma comunidade ativa e colaborativa, que disponibiliza tutoriais, cursos, assets e suporte técnico para os desenvolvedores [Unity Technologies 2020]. A Unity pode ser usada como uma ferramenta de ensino e aprendizagem de programação, pois permite criar projetos de forma visual e dinâmica, usando uma interface gráfica e um editor de scripts integrado. A Unity também possibilita o desenvolvimento do pensamento computacional, que é a capacidade de resolver problemas usando conceitos e técnicas da ciência da computação [Wing 2006] [Guzdial 2008].

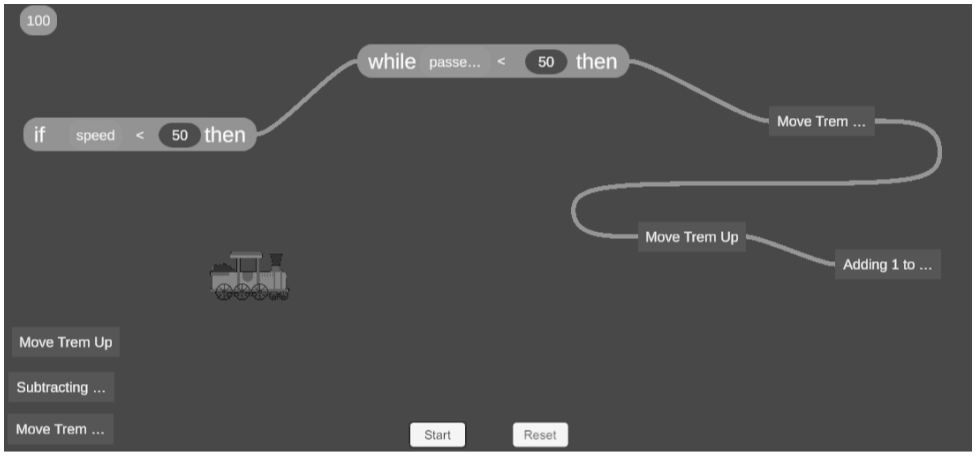
O ensino de programação baseado na Unity pode apresentar algumas vantagens, como o envolvimento e a diversão dos estudantes, a aplicação prática e contextualizada dos conceitos e estruturas de programação, a exploração de diferentes áreas e gêneros de jogos, a integração de diferentes mídias e recursos, e a possibilidade de publicação e compartilhamento dos projetos. No entanto, esse tipo de ensino também pode apresentar algumas desvantagens, como a complexidade e a curva de aprendizagem da ferramenta, a necessidade de conhecimento prévio de linguagens textuais, a dependência de recursos externos e de licenças, e a dificuldade de avaliação e de feedback [Rogers 2014, Lischka 2016].

3. Concepção, Desenvolvimento e Avaliação da Ferramenta/Plugin

Nesta seção, será descrita a concepção, o desenvolvimento e a avaliação da ferramenta/plugin para Unity que permite o ensino de programação de forma educativa, por meio de uma interface de codificação visual em blocos em tempo de execução.

3.1. Concepção

A concepção da ferramenta/plugin foi baseada em uma revisão da literatura sobre métodos eficientes para ensinar programação, especialmente a abordagem de codificação visual, que consiste em usar elementos gráficos, como blocos, ícones, cores e formas, para representar e manipular elementos de programação, como variáveis, operadores, funções e estruturas de controle [Berglund and Wiggberg 2018] [Moreno-León et al. 2015]. A partir dessa revisão, foi feito um processo de brainstorming individual, que é uma alternativa para o método convencional que permite uma maior liberdade em escolhas, para decidir a forma de implementação usando essas metodologias. O produto final, visualmente falando, é a geração de um fluxograma interativo que simula uma programação em blocos, assim temos blocos de condicionais (IF, ELSE) e laços de repetições (WHILE, FOR), além de blocos de ações (blocos que agem sobre determinado objeto) e variáveis.



[Figura 1: Exemplo de fluxograma gerado pela ferramenta/plugin]

3.2. Desenvolvimento

O desenvolvimento da ferramenta/plugin foi dividido em quatro etapas, que serão descritas a seguir.

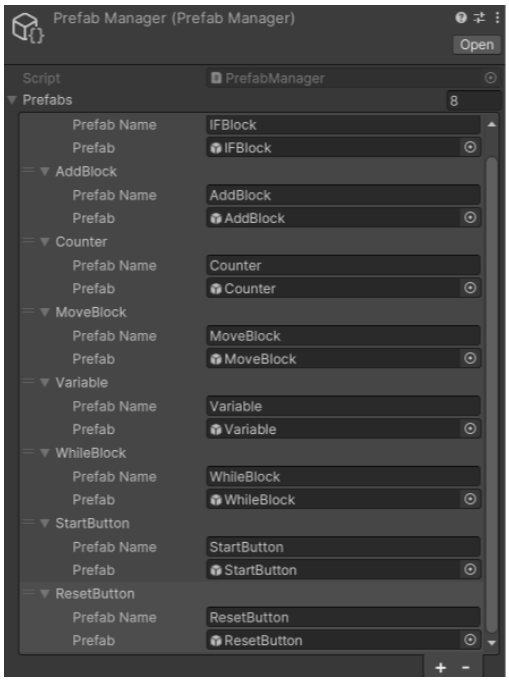
3.2.1 Programação da lógica dos blocos de programação

A primeira etapa foi a programação da lógica dos blocos de programação, que são os blocos que representam as estruturas de controle, como condicionais (IF, ELSE) e laços de repetição (WHILE, FOR). Para isso, foi usado o paradigma de programação orientada a objetos, que consiste em definir classes que representam os conceitos e as características dos blocos, e criar objetos que são as instâncias dessas classes. Cada classe de bloco possui um método chamado Execute(), que é responsável por executar a ação específica do bloco, como avaliar uma expressão, executar um bloco interno ou seguir para o próximo bloco. Cada classe de bloco também possui atributos que armazenam as informações necessárias para o seu funcionamento, como os objetos que representam os blocos seguintes, as variáveis que armazenam os valores a serem comparados, e os valores que definem a curvatura dos caminhos que conectam os blocos.

Para implementar o polimorfismo, que é a capacidade de um objeto assumir diferentes formas, foi usado o conceito de interface, que é um conjunto de métodos que uma classe deve implementar. Assim, foi definida uma interface chamada IExecutable, que possui o método Execute(), e todas as classes de blocos de programação implementam essa interface. Dessa forma, é possível tratar todos os blocos de forma genérica, sem se preocupar com os detalhes de cada um, e invocar o método Execute() de forma dinâmica, de acordo com o tipo do bloco.

Para criar a interface gráfica dos blocos de programação, foi usado o conceito de prefab, que é um modelo de objeto que pode ser reutilizado em diferentes cenas. Assim, foi criado um prefab para cada tipo de bloco, que possui um componente de imagem, que define a aparência do bloco, e um componente de script, que define o comportamento do bloco. O componente de script é uma classe que herda da classe MonoBehaviour, que é a classe base para todos os scripts que interagem com o ambiente da Unity. O componente de script também possui uma referência para a classe

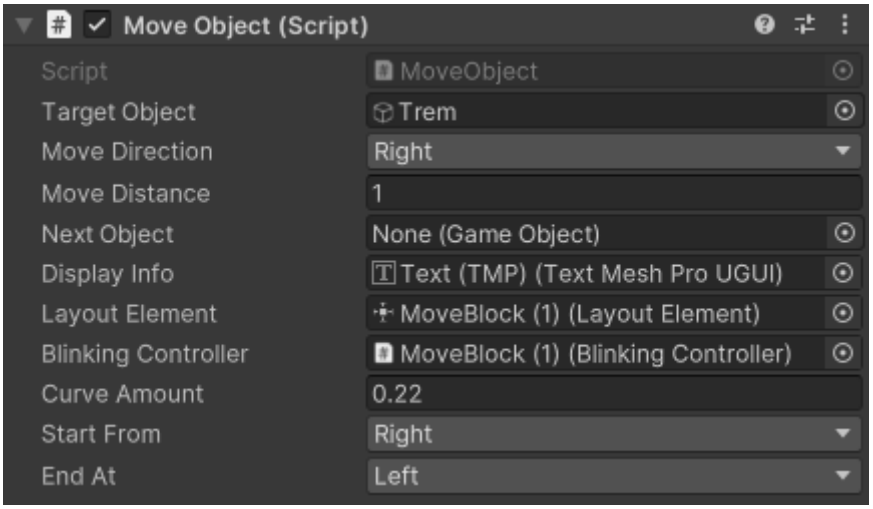
de bloco correspondente, que é a classe que implementa a lógica do bloco. A figura 2 mostra o gerenciador de prefabs, contendo todos os objetos disponíveis para criação, além de permitir adicionar ou remover mais prefabs.



[Figura 2: Gerenciador de prefabs, usando Scriptable Objects]

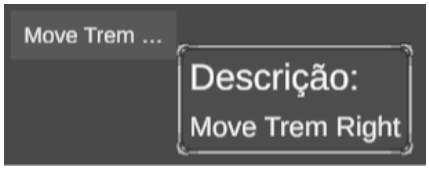
3.2.2 Programação dos blocos de ação

A segunda etapa foi a programação dos blocos de ação, que são os blocos que interagem com um objeto alvo, como movimentar, rotacionar, escalar, mudar de cor, etc. Para isso, foi usado o mesmo paradigma, conceito e estrutura da etapa anterior, ou seja, foi definida uma classe para cada tipo de bloco de ação, que implementa a interface IExecutable e possui o método Execute(), que realiza a ação específica do bloco sobre o objeto alvo. Cada classe de bloco de ação também possui atributos que armazenam as informações necessárias para o seu funcionamento, como o objeto alvo, os valores dos parâmetros da ação, e os objetos que representam os blocos seguintes.



[Figura 3: Configurações do bloco de movimentação no inspetor]

Para criar a interface gráfica dos blocos de ação, foi usado o mesmo conceito e estrutura da etapa anterior, ou seja, foi criado um prefab para cada tipo de bloco de ação, que possui um componente de imagem e um componente de script, que possui uma referência para a classe de bloco correspondente. A figura 4 mostra um exemplo de prefab do bloco de movimentação, que possui um componente de imagem e um componente de script chamado MoveObject.



[Figura 4: Exemplo de prefab do bloco de movimentação]

Os blocos de ação implementados para teste foram um para incrementar ou reduzir o valor de um contador ou propriedade do objeto alvo, e outro para movimentar o objeto em uma das quatro direções (Cima, baixo, esquerda, direita).

3.2.3 Programação dos blocos de variável

A terceira etapa foi a programação dos blocos de variável, que são os blocos que representam as propriedades dos objetos alvo, como posição, rotação, escala, cor, etc. Para isso, foi usado o mesmo paradigma, conceito e estrutura das etapas anteriores, ou seja, foi definida uma classe para o bloco de variável, que implementa a interface IExecutable e possui o método Execute(), que retorna o valor da propriedade do objeto alvo. A classe de bloco de variável também possui atributos que armazenam as informações necessárias para o seu funcionamento, como o objeto alvo, o nome da propriedade, o tipo da propriedade, e os objetos que representam os blocos seguintes.

Para criar a interface gráfica do bloco de variável, foi usado o mesmo conceito e estrutura das etapas anteriores, ou seja, foi criado um prefab para o bloco de variável, que possui um componente de imagem e um componente de script, que possui uma referência para a classe de bloco correspondente.

A princípio, essas variáveis representam as propriedades que foram adicionadas com o script gerado na etapa seguinte, então elas fazem referência a alguma propriedade do GameObject escolhido como alvo. De forma responsiva, ao adicionar ou remover propriedades, essas mudanças vão ser refletidas no bloco de variável, que vai mostrar o nome, o tipo e o valor da propriedade. Junto ao bloco de variável, foi criado um novo tipo de variável para servir como contador genérico, que pode ser usado de diversas formas, como, por exemplo, para simular a funcionalidade de um ForLoop em um fluxograma.

3.2.4 Programação da parte gráfica e de UX

A quarta e última etapa foi a programação da parte gráfica e de UX, que são os aspectos visuais e de interação da ferramenta/plugin. Para isso, foram usados os conceitos e as ferramentas da Unity, como o Canvas, que é um componente que permite criar e organizar elementos de interface de usuário, como botões, textos, imagens, etc., e o EventSystem, que é um componente que permite gerenciar e responder aos eventos de entrada, como cliques, arrastos, teclas, etc. A figura 11 mostra um diagrama que ilustra a estrutura dos componentes gráficos e de UX da ferramenta/plugin.

Para criar a interface gráfica da ferramenta/plugin, foram usados os seguintes elementos:

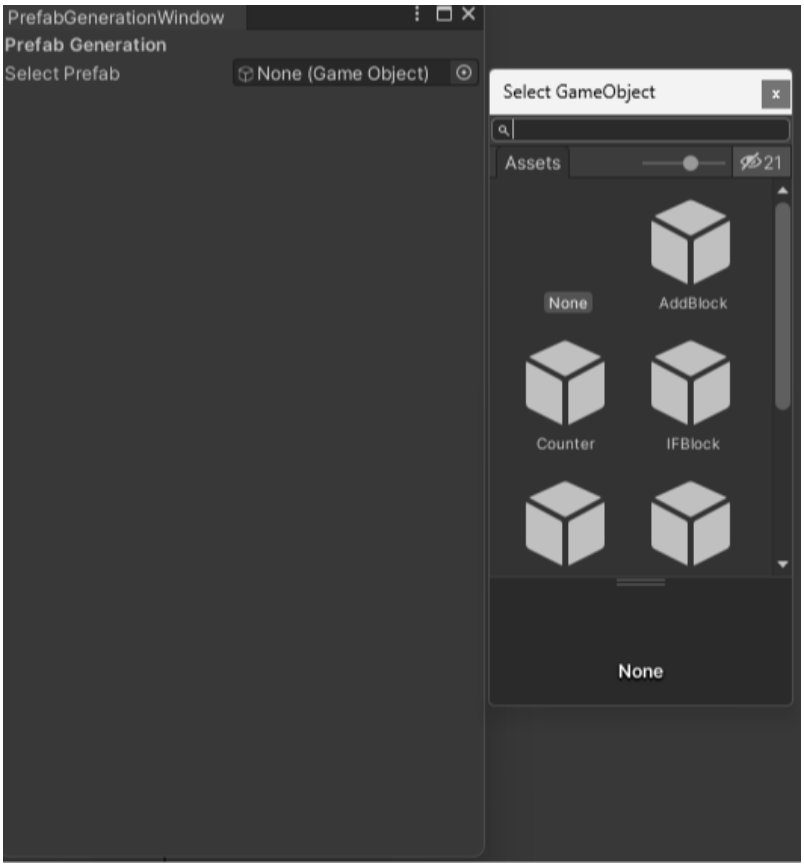
- Um canvas, que é um elemento que serve como fundo e contêiner para outros elementos, que ocupa toda a tela.
- Um bloco de IF, que é um elemento que serve para criar um bloco condicional IF, que tem uma imagem do bloco e um texto “IF”.
- Um bloco de WHILE, que é um elemento que serve para criar um bloco de repetição WHILE, que tem uma imagem do bloco e um texto “WHILE”.
- Um bloco de ação de incremento, bloco usado para aumentar ou diminuir a quantidade de um contador ou propriedade do objeto alvo.
- Um bloco de movimentação, usado para movimentar o objeto alvo em uma das quatro direções básicas.
- Um bloco de variável, que é um elemento que serve para criar um bloco de variável, usado para armazenar o valor de determinada propriedade do objeto alvo.
- Um bloco de variável do tipo contador.
- Imagem para representar o objeto alvo.

Para criar a interação da ferramenta/plugin, foram usados os seguintes recursos:

- Um sistema de arrastar e soltar, que permite ao usuário mover e reorganizar os blocos da lista do fluxograma, usando o mouse.
- Um sistema de encaixe, que permite ao usuário conectar os blocos uns aos outros, de acordo com as regras de compatibilidade, através dos diferentes tipos de cliques.
- Um sistema de edição, que permite ao usuário modificar os valores e as propriedades dos blocos, usando campos de texto, botões, menus customizados.

- Um sistema de execução, que permite ao usuário iniciar e resetar a execução do fluxograma, usando os botões de Play e Reset, e uma animação que mostra o bloco que está sendo executado e o valor das variáveis.
- Um sistema ToolTip para detalhar mais informações acerca de um bloco quando o usuário passa o mouse por cima do objeto.
- Sub-menus para facilitar a criação de alguns elementos do sistema, como prefabs e scripts de propriedades.

  A figura 5 mostra um exemplo de interface gráfica de submenu implementada com a API do Unity para o sistema.



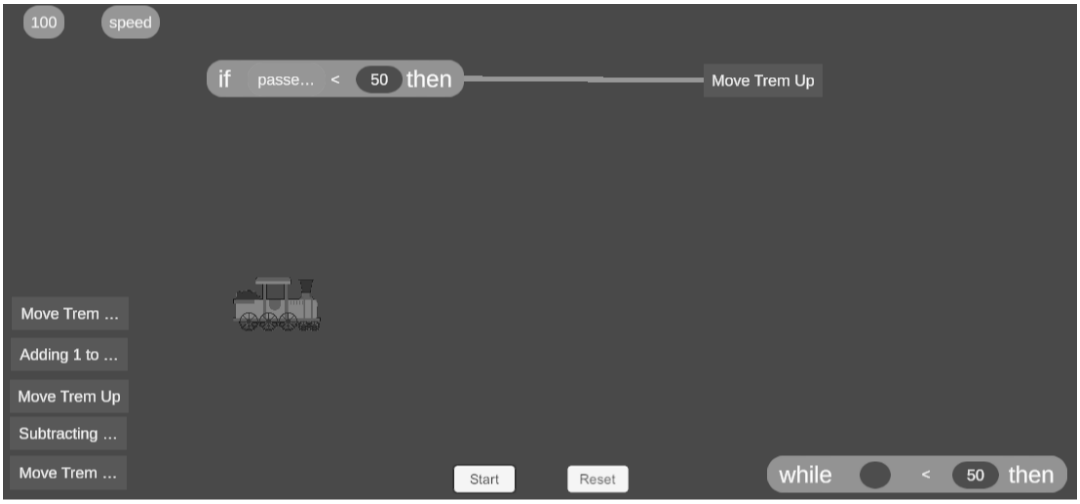
[Figura 5: Interface gráfica da ferramenta/plugin, submenu para geração de prefabs na cena]

3.4 Avaliação

A avaliação da ferramenta/plugin foi feita com o objetivo de verificar a sua usabilidade e a sua eficácia para o ensino de programação. Para isso, foram usados os seguintes métodos e critérios de avaliação:

- Um teste qualitativo com cenários, que consiste em observar e analisar o comportamento e as reações dos usuários ao usar a ferramenta/plugin para realizar tarefas específicas, relacionadas à criação e à execução de fluxogramas. O teste qualitativo com cenários foi feito com um grupo de 25 estudantes do

ensino médio, que possuem pouco ou nenhum conhecimento prévio de programação, usando um ambiente realista e um conjunto de roteiros, que envolvem dois cenários diferentes. O primeiro cenário consiste em criar um projeto com um objeto alvo “Trem”, e implementar um fluxograma com um bloco condicional IF, onde a condição é o número de passageiros do trem, e a ação é mover o trem uma determinada posição e parar, caso a condição seja verdadeira. O segundo cenário consiste em criar um projeto com um objeto alvo “Carro”, e implementar um fluxograma com um bloco de repetição WHILE, onde a condição é a velocidade do carro, e a ação é aumentar a velocidade do carro até atingir um determinado valor. A figura 6 mostra um exemplo de fluxograma para o primeiro cenário.



[Figura 6: Exemplo de fluxograma para o primeiro cenário]

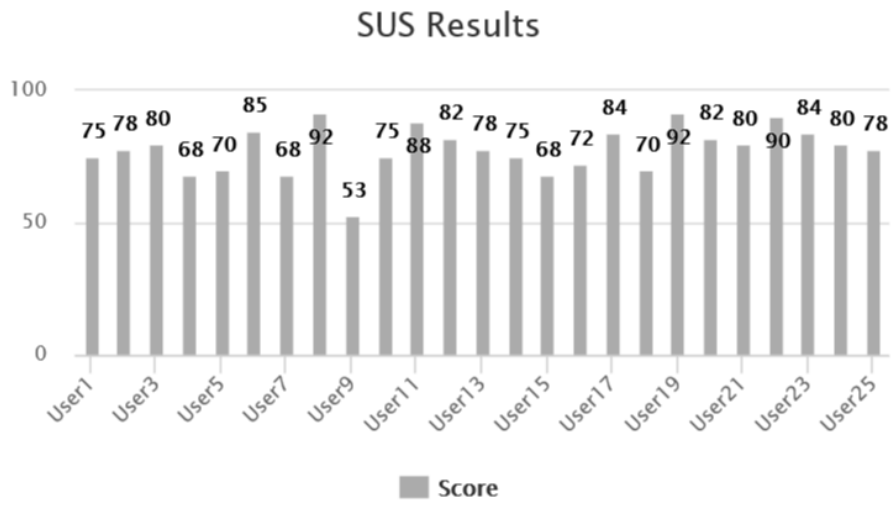
O resultado do teste qualitativo com cenários foi analisado por meio de uma observação direta, mas a fim de retirar o melhor do teste qualitativo, o mesmo foi pareado com um teste de usabilidade para se entender a eficiência da ferramenta.

- Um teste de usabilidade, que consiste em verificar se a ferramenta/plugin é fácil de usar, de aprender e de entender, sem causar confusão, frustração ou desmotivação nos usuários. O teste de usabilidade foi feito com o mesmo grupo de 25 estudantes do ensino médio, que participaram do teste qualitativo com cenários, usando o mesmo ambiente e o mesmo conjunto de roteiros. O resultado do teste de usabilidade foi medido por meio de um questionário, que avalia a satisfação, a facilidade, a clareza dos usuários em relação à ferramenta/plugin, usando a escala SUS (System Usability Scale), que é um método simples e confiável para medir a usabilidade de um sistema [Brooke 1996]. A escala SUS consiste em 10 itens com cinco opções de resposta, que variam de Discordo totalmente a Concordo totalmente. O questionário usado no teste de usabilidade está mostrado na figura 7.

1. Eu acho que gostaria de usar esse sistema com frequência.
2. Eu acho o sistema desnecessariamente complexo.
3. Eu achei o sistema fácil de usar.
4. Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.
5. Eu acho que as várias funções do sistema estão muito bem integradas.
6. Eu acho que o sistema apresenta muita inconsistência.
7. Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente.
8. Eu achei o sistema atrapalhado de usar.
9. Eu me senti confiante ao usar o sistema.
10. Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.

[Figura 7: Exemplo de questionário usado no teste de usabilidade]

O resultado do teste de usabilidade foi positivo, pois a ferramenta/plugin obteve uma média geral de 77,48 na escala SUS, o que indica um nível bom de usabilidade.



[Figura 8: Resultado do teste de usabilidade por usuário]

Após a realização dos testes, foram coletados feedbacks dos usuários sobre o uso, a compreensão e a eficácia da ferramenta/plugin, por meio de uma entrevista aberta, que permite aos usuários expressarem suas opiniões, sugestões e críticas sobre a ferramenta/plugin. Os feedbacks dos usuários foram analisados e ajustados conforme o necessário, para melhorar a qualidade e a adequação da ferramenta/plugin. A tabela 1 mostra alguns exemplos de feedbacks dos usuários sobre a ferramenta/plugin.

[Tabela 1: Feedbacks dos usuários sobre a ferramenta/plugin]

Participantes	Feedback
User9	“Não entendi muito bem o que tinha que fazer, precisei de ajuda para entender como movimentar e ligar os blocos, mas depois consegui pegar o jeito ficou intuitivo.”
User17	“Achei uma ideia interessante, entendo muito pouco de programação, uma resposta gráfica fica mais fácil de entender o que cada coisa faz, só faltou algumas confirmações visuais em umas etapas.”
User21	“Consegui aprender o básico, ela é bem amigável e tem uma estrutura modular, o que gostei é que você interage ativamente com ela, no lugar de ficar digitando códigos e textos. Sobre o visual, ele é atraente, mas acho que dá pra melhorar.”

5. Considerações Finais e Sugestões para Trabalhos Futuros

Nesta seção, serão apresentadas as considerações finais e as sugestões para trabalhos futuros sobre a ferramenta/plugin para Unity que permite o ensino de programação de forma educativa, por meio de uma interface de codificação visual em blocos em tempo de execução.

5.1 Considerações Finais

O objetivo deste trabalho foi desenvolver uma ferramenta/plugin para Unity que permitisse o ensino de programação de forma educativa, por meio de uma interface de codificação visual em blocos em tempo de execução. Para isso, foi realizada uma revisão da literatura sobre os conceitos, as técnicas e os modelos de programação, bem como sobre as abordagens e as ferramentas para o ensino de programação, especialmente a codificação visual, que consiste em usar elementos gráficos, como blocos, ícones, cores e formas, para representar e manipular elementos de programação, como variáveis, operadores, funções e estruturas de controle. A partir dessa revisão, foi feito um processo de brainstorming individual, que é uma alternativa para o método convencional que permite uma maior liberdade em escolhas, para decidir a forma de implementação usando essas metodologias. O produto final, visualmente falando, é a geração de um fluxograma interativo que simula uma programação em blocos, assim temos blocos de condicionais (IF, ELSE) e laços de repetições (WHILE, FOR), além de blocos de ações (blocos que agem sobre determinado objeto) e variáveis.

O desenvolvimento da ferramenta/plugin foi dividido em quatro etapas, que foram a programação da lógica dos blocos de programação, a programação dos blocos de ação, a programação dos blocos de variável e a programação da parte gráfica e de UX. Para isso, foram usados os conceitos e as ferramentas da Unity, como o paradigma de programação orientada a objetos, o conceito de interface, o conceito de prefab, o componente Canvas, o componente EventSystem, e o menu customizado. O resultado

do desenvolvimento foi uma ferramenta/plugin funcional, que permite ao usuário criar e executar fluxogramas simples e complexos, usando blocos de diferentes tipos, cores e tamanhos, e caminhos que conectam os blocos, que interagem com um objeto alvo, como um trem ou um carro, que se movimenta, rotaciona, escala, muda de cor, etc.

A avaliação da ferramenta/plugin foi feita com o objetivo de verificar a sua usabilidade e a sua eficácia para o ensino de programação. Para isso, foram usados os seguintes métodos e critérios de avaliação: um teste qualitativo com cenários, que consiste em observar e analisar o comportamento e as reações dos usuários ao usar a ferramenta/plugin para realizar tarefas específicas, relacionadas à criação e à execução de fluxogramas; um teste de usabilidade, que consiste em verificar se a ferramenta/plugin é fácil de usar, de aprender e de entender, sem causar confusão, frustração ou desmotivação nos usuários, usando a escala SUS (System Usability Scale), que é um método simples e confiável para medir a usabilidade de um sistema; O resultado da avaliação foi positivo, pois a ferramenta/plugin obteve um alto nível de usabilidade e de eficácia, de acordo com as medidas e os feedbacks dos usuários.

O trabalho realizado atingiu o objetivo proposto. A ferramenta/plugin demonstrou ser uma alternativa viável e interessante para o ensino de programação, pois combina os benefícios da codificação visual, como a facilidade, a clareza e a diversão, com os benefícios da Unity, como a flexibilidade, a interatividade e a criatividade. A ferramenta/plugin também contribuiu para o desenvolvimento de habilidades e competências relacionadas à programação, como o pensamento computacional, a lógica, a abstração, a decomposição, a generalização e a depuração. A ferramenta/plugin pode ser usada por professores e alunos de diferentes níveis de ensino, desde o ensino fundamental até o ensino superior, para aprender e ensinar programação de forma lúdica e motivadora. Ela também pode ser usada por desenvolvedores e entusiastas de jogos, para criar e testar protótipos de jogos, usando a codificação visual em blocos em tempo de execução. A ferramenta/plugin representa um avanço na área de ensino de programação, pois integra conceitos, técnicas e ferramentas de programação com conceitos, técnicas e ferramentas de jogos, criando uma experiência de aprendizagem única e inovadora.

5.2 Sugestões para Trabalhos Futuros

A ferramenta/plugin desenvolvida neste trabalho apresenta algumas limitações e possibilidades de melhoria, que podem ser exploradas em trabalhos futuros. Algumas sugestões para trabalhos futuros são:

- Melhorar a experiência e a interação do usuário com a ferramenta/plugin, por meio de um design mais atraente, intuitivo e personalizável, e que ofereça feedbacks visuais e sonoros mais claros e consistentes, que indiquem o estado, o progresso e o resultado da execução do fluxograma.
- Explorar mais a API do Unity, para fazer um sistema mais responsivo e polido, que aproveite os recursos e as funcionalidades da plataforma, como a renderização, a animação, a física, o som, a rede, etc., e que se adapte a diferentes dispositivos, resoluções e plataformas, como desktop, mobile, web, etc.
- Adicionar mais tipos de blocos de ação e de variável, que permitam ao usuário interagir com mais aspectos do objeto alvo, como o som, a luz, a câmera, o

teclado, o mouse, o toque, etc., e que permitam ao usuário criar e manipular mais tipos de dados, como strings, arrays, listas, dicionários, etc.

- Ampliar o escopo e a complexidade dos cenários e das avaliações, que desafiem o usuário a resolver problemas mais reais e interessantes, que envolvam conceitos e estruturas de programação mais avançados, como funções, classes, objetos, herança, polimorfismo, etc., e que avaliem o desempenho, a criatividade e a colaboração do usuário, usando métodos e critérios mais sofisticados, como rubricas, portfólios, peer review, etc.

Referências

Alves, A. A. (2013) "Uma abordagem baseada em jogos para o ensino de programação," Dissertação de Mestrado, Universidade do Minho, Braga, Portugal.

Bau, D., Zhou, B., Khosla, A., Oliva, A. and Torralba, A. (2017) "Network dissection: Quantifying interpretability of deep visual representations," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6541-6549.

Berglund, A. and Wiggberg, M. (2018) "The Conflicting Paradigms of Programming Education: A Review and a Proposed Approach", International Journal of Engineering Education, vol. 34, no. 6, p. 1863-1878.

Brooke, J. (1996) SUS: a quick and dirty usability scale. In: JORDAN, P. W.; THOMAS, B.; WEERDMEESTER, B. A.; MCCLELLAND, I. L. (Ed.). Usability evaluation in industry. London: Taylor and Francis, p. 189-194.

Gomes, A. and Mendes, A. J. (2017) "Learning to Program - Difficulties and Solutions", International Journal of Engineering Pedagogy, vol. 7, no. 2, p. 72-87.

Guzdial, M. (2008) "Education: paving the way for computational thinking," Communications of the ACM, vol. 51, no. 8, pp. 25-27.

Kelleher, C. and Pausch, R. (2005) "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers," ACM Computing Surveys, vol. 37, no. 2, pp. 83-137.

Lischka, J. A. (2016) "Review: Innovators in Digital News by Lucy Küng," The Journal of Media Innovations, vol. 3, no. 2, pp. 88-90.

Maloney, J., Resnick, M., Rusk, N., Silverman, B. and Eastmond, E. (2010) "The Scratch programming language and environment," ACM Transactions on Computing Education (TOCE), vol. 10, no. 4, p. 16.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., Laxer, C., Thomas, L., Utting, I. and Wilusz, T. (2001) "A multi-national, multi-institutional study of assessment of programming skills of first-year CS students," SIGCSE Bulletin, vol. 33, no. 4, pp. 125-180.

Moreira, A. F. B. (2016) "Uma ferramenta para o ensino de programação baseada em blocos," Trabalho de Conclusão de Curso, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil.

Moreno-León, J., Robles, G. and Román-González, M. (2015) "A game-based approach to the teaching of object-oriented programming languages", In: Proceedings of the

15th Koli Calling Conference on Computing Education Research, Koli, Finland, p. 167-168.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. and Kafai, Y. (2009) “Scratch: programming for all,” Communications of the ACM, vol. 52, no. 11, pp. 60-67.

Rogers, P. (2014) “Theory of change,” Methodological Briefs: Impact Evaluation 2, UNICEF Office of Research, Florence.

Sáez-López, J. M., Román-González, M. and Vázquez-Cano, E. (2016) “Visual programming languages integrated across the curriculum in elementary school: A two year case study using Scratch in five schools”, Computers & Education, vol. 97, p. 129-141.

Sorva, J., Karavirta, V. and Malmi, L. (2013) “A review of generic program visualization systems for introductory programming education,” ACM Transactions on Computing Education, vol. 13, no. 4, pp. 15:1-15:64.

Unity Technologies (2020) “Unity - Manual: Unity User Manual (2020.3 LTS),” [Online]. Available: https://docs.unity3d.com/Manual/index.html.

Van Roy, P. and Haridi, S. (2004) Concepts, Techniques, and Models of Computer Programming. MIT Press.

Weintrop, D. and Wilensky, U. (2015) “Defining computational thinking for mathematics and science classrooms,” Journal of Science Education and Technology, vol. 25, no. 1, pp. 127-147.

Wing, J. M. (2006) “Computational thinking,” Communications of the ACM, vol. 49, no. 3, pp. 33-35.