FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS – FATECS
CURSO

WALLACE REYLAN DE LIRA SOARES
21505399

METODOLOGIA DE ENSINO DE PROGRAMAÇÃO ORIENTADA AO DESENVOLVIMENTO DE UM PROJETO DE JOGO DIGITAL COM UNITY 2D

BRASÍLIA
2020

WALLACE REYLAN DE LIRA SOARES

METODOLOGIA DE ENSINO DE PROGRAMAÇÃO ORIENTADA AO DESENVOLVIMENTO DE UM PROJETO DE JOGO DIGITAL COM UNITY 2D

Trabalho de Conclusão de Curso (TCC) apresentado como um dos requisitos para a conclusão do curso de Engenharia de Computação do UniCEUB– Centro Universitário de Brasília

Orientador (a)**: Prof. MsC Francisco Javier De Obaldía Díaz**

BRASÍLIA
2020

WALLACE REYLAN DE LIRA SOARES

METODOLOGIA DE ENSINO DE PROGRAMAÇÃO ORIENTADA AO DESENVOLVIMENTO DE UM PROJETO DE JOGO DIGITAL COM UNITY 2D

Trabalho de Conclusão de Curso (TCC) apresentado como um dos requisitos para a conclusão do curso de Engenharia de Computação do UniCEUB – Centro Universitário de Brasília

Orientador (a)**: Prof. MsC Francisco Javier De Obaldía Díaz**

Brasília, 2020.

BANCA EXAMINADORA

_____

Nome e titulação.
Orientador (a)

_____

Nome e titulação.
Examinador (a)

_____

Nome e titulação.
Examinador (a)

METODOLOGIA DE ENSINO DE PROGRAMAÇÃO ORIENTADA AO DESENVOLVIMENTO DE UM PROJETO DE JOGO DIGITAL COM UNITY 2D

PROGRAMMING TEACHING METHODOLOGY GUIDED TO A DIGITAL GAME PROJECT DEVELOPMENT WITH UNITY 2D

Wallace Reylan de Lira Soares [1], Francisco Javier de Obaldía Díaz [2], William Roberto Malvezzi [3], Flávio Antonio Klein [4]

Resumo: A programação se mostrou ao longo dos anos uma habilidade poderosa, abrindo portas para o desenvolvimento e estimulando raciocínio lógico e solução de problemas. Este artigo propõe uma metodologia de ensino engajadora e objetiva detalhada em quatro encontros e fundamentada na projeção e criação de um jogo, onde alunos de nível médio a superior podem aprender conceitos de lógica de programação, colisões físicas, máquinas de estado e linguagem C# enquanto se divertem durante a condução de seu desenvolvimento. O jogo proposto segue o gênero de plataforma 2D, desenvolvido na engine Unity de maneira prática, trabalhando as principais mecânicas deste estilo de jogo, sua análise de efetividade e aplicação como ferramenta de ensino foi feita baseada na apresentação final dos discentes como resultados práticos e pesquisas teóricas em modelo questionário. Demonstrando conceitos e seu desenvolvimento ao final do minicurso, bem como o sucesso dos fatores de motivação, aprendizado e desenvolvimento alcançados com o método.
Palavras-chave: Ensino. Programação. Jogos digitais. Unity.

Abstract: Programming has proven to be a mighty skill over the years, opening doors for development and stimulating logical reasoning and problem solving. This article proposes an engaging and objective teaching methodology detailed in four meetings and based on the projection and creation of a game, where students from middle to higher level can learn concepts of programming logic, physical collisions, state machines and C # language while having fun during their development. The proposed game follows the 2D platform genre, developed on the Unity engine in a practical way, working on the main mechanics of this game style, its analysis of effectiveness and application as a teaching tool was made based on the final presentation of students as practical results, and theoretical research in a questionnaire model. Demonstrating concepts and their development at the end of the short course, as well as the success of the motivation, learning and development factors achieved with the method.
Keywords: Teaching. Programming. Digital games. Unity.

[1] UniCEUB, aluno.
[2] UniCEUB, orientador.
3 UniCEUB, primeiro examinador.
[4] UniCEUB, segundo examinador.

[2]UniCEUB – Centro Universitário de Brasília

1 INTRODUÇÃO

É natural que o estudo da programação e a busca por criações digitais seja crescente. Contudo, Valaski e Paraiso (2012) abordam que é comum que as disciplinas de programação apresentam um alto fator de reprovação nos cursos de computação, o qual muitas vezes é o primeiro contato do aluno. A disciplina introdutória de programação de computadores não é abordada como uma atividade-fim, assim, os estudantes apresentam baixa motivação em aprender o conteúdo proposto (Carvalho et al, 2016).

Um meio para contrapor esse fator, é a aplicação de metodologias de ensino mais amigáveis para os conceitos iniciais de programação. Segundo Paula et al (2009) construir representações mentais que abstraiam completamente os problemas é uma tarefa difícil para os alunos, sendo uma responsabilidade dos educadores desenvolverem meios que direcionam os alunos para essa capacidade.

Em concordância Hoed (2016) afirma que é importante que o ensino de programação seja prazeroso e trate de situações reais e dinâmicas para envolver o discente. Assim, este artigo tem como objetivo propor uma metodologia de ensino de programação e conceitos computacionais que compõem a base dos cursos de tecnologia, aplicada a utilização prática destes conceitos por alunos através do desenvolvimento de um projeto de jogo digital desenvolvido na Unity (plataforma de desenvolvimento de jogos), a fim de manter os alunos engajados, trazer ludicidade e desafios mais visuais ao aprendizado.

2 REVISÃO BIBLIOGRÁFICA

2.1 Ensino de programação engajador e objetivo

Diversos estudos buscam compreender as dificuldades de aprendizado. Gomes et al em 2016 discute as barreiras e razões pelas quais os alunos iniciantes se sentem descontentes com suas experiências de introdução na programação. Propostas com o intuito de tornar a aprendizagem de programação mais divertida e empolgante têm sido feitas utilizando jogos como elemento potencialmente motivador (Costa e Rocha, 2018). Trabalhar a criação de jogos não traz apenas empolgação. Conforme Carosia et al (2017) Programar um jogo e acompanhar seus resultados de forma visual é desafiador, é capaz de motivar o aluno a resolver problemas. Com isso, sendo uma abordagem a ser tratada para soluções de problemas. O ensino proposto por Falcão e Júnior (2015), demonstrou em um workshop com desenvolvimento de jogos uma absoluta participação e motivação dos alunos. Em uma outra oficina realizada em 2011 por Marques, foi observado em pesquisa com os participantes que houve aumento de interesse na área de informática entre todos os alunos, além de apontarem positivamente a utilização de jogos como fator motivacional.

2.2 Projetos práticos

O ensino de programação não apresenta simplicidade. Conforme Coutinho et al (2018) ensinar lógica de programação e codificação são tarefas complexas, sendo um desafio ao aluno relacionar teoria com a prática. A prática precisa de uma abordagem que enalteça a teoria e esteja próxima do universo do aluno. Grotta e Prado (2018) destacam que a aprendizagem por projetos segue a diretriz de maximizar a performance do ensino por meio do fazer, sendo este, uma aplicação próxima à vida do estudante. A produção de um jogo digital está alinhada com a proposta de projeto e reafirma a proximidade com o mundo do discente, segundo Andrade et al (2016) “Os alunos afirmaram que a oportunidade de criar um jogo do início, apesar dos desafios, é extremamente motivador por ver o funcionamento do produto final”, produto este, que ilustra ludicamente os conceitos aplicados na prática.

2.3 Unity engine no contexto educacional

A Unity é uma ferramenta para desenvolvimento de jogos de fácil acesso para iniciantes. Segundo Unity (2020) sua aplicação fornece componentes de física e renderização que reduzem a complexidade do projeto (Comber et al, 2019). A Unity é adequada para a sala de aula, uma vez que também é gratuita e amplamente utilizada, desta forma, possui diversos fóruns para sanar as mais variadas dúvidas (Falcão e Júnior, 2015).

3 METODOLOGIA DO TRABALHO

A abordagem de ensino desenvolvida neste artigo foi de uma pesquisa de natureza aplicada em encontros semanais, com duração de 1 hora e meia a 2 horas em um contexto online e ao vivo, através da ferramenta Google Meet. Proposto com base em sua pesquisa bibliográfica, o projeto foi apresentado à escola Ctrl+Play na unidade de Taguatinga-Norte, que cedeu 5 estudantes os quais tiveram uma experiência introdutória em programação com Python.

3.1 Planejamento do minicurso

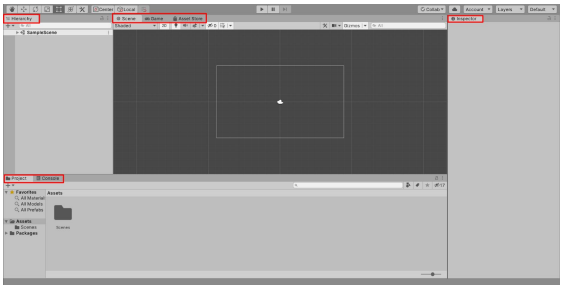
São diversos os motivos para se aplicar o desenvolvimento de jogos como ferramenta de estudo. Projetar um jogo requer planejamento, criatividade, engajamento, participação ativa, estratégia, trabalho em equipe e outros. O projeto utiliza na prática, conceitos comuns a cursos de computação, como programação em script, noções de orientação a objeto, vetores, conceitos básicos de física e máquinas de estado, visando proporcionar uma base firme nestes conceitos para alunos que não tiveram contato anterior com os mesmos, ou consolidar tais ideias dentro de um contexto mais visual e tangível para os que já tiveram.

3.2 Execução do minicurso

3.2.1 Primeiro encontro

A primeira parte do encontro foi para apresentar a ferramenta e debater sobre a proposta do jogo a ser desenvolvido. Os participantes conversaram para decidir e planejar as funcionalidades que gostariam de criar com base nas artes disponíveis. A segunda parte foi focada na familiarização com a Unity, os alunos conheceram as janelas principais da ferramenta, conforme apresentado na Figura 1.

Figura 1: Captura de tela do Ambiente de desenvolvimento da Unity



Fonte: Acervo próprio, 2020

Hierarchy – A hierarchy é a janela que guarda a referência de todos os objetos presentes no jogo, é com ela que são inseridos e organizados.

Inspector – O inspector disponibiliza todas as informações dos componentes daqueles objetos ou arquivos selecionados. Essa janela é utilizada para fazer certas modificações e adições de novos componentes.

Project – Aqui ficam guardados todos os arquivos referentes ao seu projeto que serão utilizados.

Console – Janela responsável por enviar mensagens de erro, notificações e logs para auxiliar com os testes e correções.

Scene – Monta-se a cena da aplicação, todos os objetos presentes são organizados nela de acordo com a proposta de jogo.

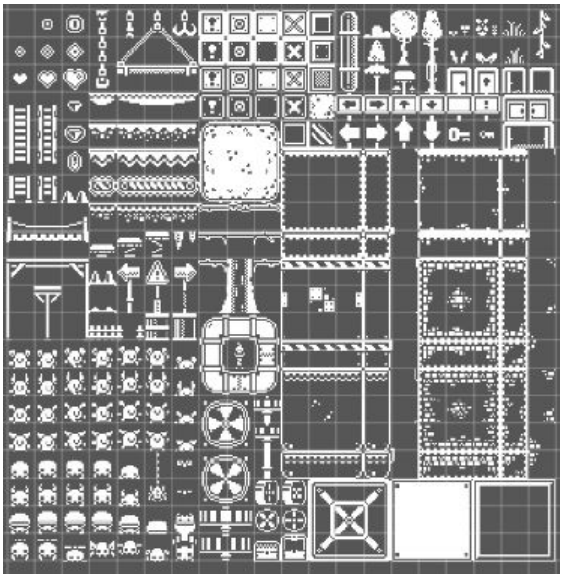
Game – Esta janela executa uma prévia da sua produção, sendo utilizada para testar etapa por etapa do desenvolvimento.

Assets Store – Em muitas produções é importante iniciar com protótipos, ou até mesmo comprar recursos prontos. Essa janela tem a função de adquirir arquivos para usar no projeto, sendo excelentes para prototipagem e úteis para adiantar etapas.

Passado este reconhecimento, organiza-se todos os arquivos necessários em pastas na

janela de Project. O próximo passo foi preparar a imagem de Tilemap (técnica de composição de jogo baseada em pequenas peças) para ser adicionada a uma nova janela chamada Tile Palette, incluída pelo caminho: Window>2D>Tile Palette, essa janela permite transformar áreas de uma imagem em pincel, para desenhar com mais liberdade o cenário do jogo, conforme a figura 2.

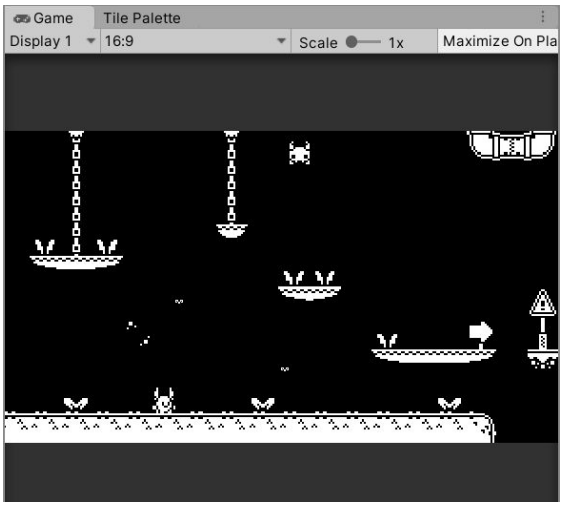
Figura 2: Captura de tela do Tilemap na janela de Tile Palette



Fonte: Acervo próprio, 2020

Boa parte das artes do jogo está presente nesta única imagem, que para utilização de cada peça é dividida em várias áreas de 16x16 pixels, para que as imagens fossem acessíveis separadamente e utilizadas no Palette. Para pintar, foi necessário adicionar dois objetos de Tilemap na aba de Hierarchy, um para desenhar chão, paredes e plataformas, e outro para desenhar os acessórios. A Figura 3 mostra a imagem de exemplo de um cenário inicial montado.

Figura 3: Captura de tela de um cenário inicial desenvolvido em aula



Fonte: Acervo próprio, 2020

3.2.2 Segundo encontro

O segundo encontro contou com a inserção do objeto do personagem na cena, e a adição dos componentes de física e colisões ao projeto. O personagem jogável deve ser capaz de se movimentar e saltar entre o chão e as plataformas do cenário, para que seja possível, o personagem precisa se comportar como se possuísse uma massa, tivesse força gravitacional agindo sobre ele e fosse capaz de acionar movimentos e forças físicas de velocidade e impulso. Programar isso tudo sem nenhuma base pronta pode ser uma etapa trabalhosa. A Unity conta com um componente de física para objetos chamado RigidBody, que no caso, foi utilizada a versão em 2D (Rigidbody 2D).

Após a adição deste componente ao objeto do jogador, o mesmo passa a sofrer automaticamente as ações da gravidade, além de ter a opção de ativar as forças físicas antes mencionadas via C# script. Ativar somente a física ainda não é suficiente, torna-se necessário também uma área de colisão para que os objetos possam interagir. Existem vários componentes de “Collider” diferentes e foi utilizado para este caso o “Circle Collider”, devido a forma redonda do personagem assim tornando o colisor mais adequado para ele.

Todos os componentes são adicionados através da janela Inspector do objeto,

clicando no botão “Add Component”. Para o objeto que guarda as plataformas, foi utilizado o “Tilemap Collider” para transformar cada peça pintada em uma peça com colisor e um “Composite Collider” para juntar os colisores que estiverem conectados, assim simplificando estas colisões.

Para a finalização parcial desta etapa, ao adicionar o “Composite Collider” entra um Rigidbody2D junto, e neste momento, o cenário vai passar a ter uma massa, para evitar que o cenário caia ou rotacione, ou seja afetado por qualquer força de outro objeto. Surge a troca do seu “Body Type” para “Static”, desta forma, ele vai desprezar as forças atuantes sobre ele. O próximo passo foi conhecer a base do Script C# para Unity.

Figura 4: Captura de tela de um C# script base da Unity

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   public class Player : MonoBehaviour
5   {
6
7       void Start()
8       {
9
10      }
11
12      void Update()
13      {
14
15      }
16  }
```

Fonte: Acervo próprio, 2020

Bibliotecas – são utilizadas por padrão sempre que criamos o script, elas permitem a utilização dos recursos e códigos da Unity.
Classes e MonoBehaviour – A classe do script carrega o nome dado a ele, assim o código é desenvolvido baseado em atributos e métodos para o objeto que for carregar aquele arquivo. MonoBehaviour é a classe principal da Unity, os scripts herdam seus métodos para implementar suas funções.
Start – Esse método é chamado uma única vez ao iniciar o projeto, ideal para carregar componentes e aplicar definições.
Update – Esse método é chamado a cada instante da execução. A lógica principal é desenvolvida nessa função, como programação de botões e interações. Conforme os conceitos se tornam mais claros, avançamos para a montagem do script de movimentação do personagem, apresentado abaixo nas figuras 5, 6 e 7.

Figura 5: Captura de tela das variáveis do script do Player

```
6   public float velocidade;
7   //esta variável é responsável
8   //pela velocidade do jogador
9   Rigidbody2D rig;
10  //Variável que vai carregar o
11  //componente de física do personagem
```

Fonte: Acervo próprio, 2020

São criadas nesta etapa duas variáveis conforme imagem 5, uma variável para o coeficiente de velocidade e outra para controlar a física. Em um primeiro momento, como recurso didático, a velocidade é definida como privada e posteriormente é alterada para pública. De modo a ser acessada diretamente pela plataforma da Unity, facilitando assim sua alteração, testes e possibilitando acesso a mesma via outros scripts.

Figura 6: Captura de tela do método start do script do Player

```
12  void Start()
13  {
14      rig = GetComponent<Rigidbody2D>();
15      //Guardamos o componente de física
16      //do personagem em nossa variável
17  }
```

Fonte: Acervo próprio, 2020

No método start apresentado na figura 6, é apresentado o GetComponent, função do MonoBehaviour da Unity responsável por acessar os componentes dos objetos. Assim, referencia se o componente de física na variável criada.

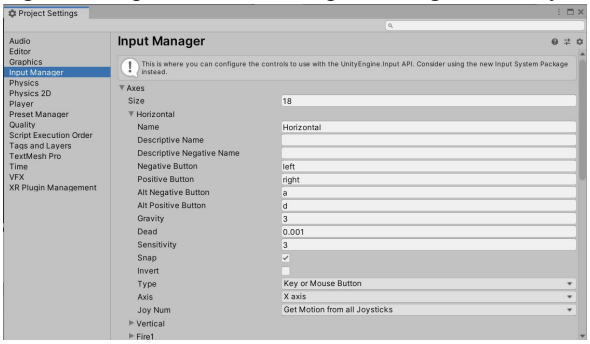
Figura 7: Captura de tela do método update do script do Player

```
20      void Update()
21      {
22          float movimento = Input.GetAxis("Horizontal");
23          //movimento vai de 0 a -1 para o botão negativo(esquerda)
24          //movimento vai de 0 a 1 para o botão positivo(direita)
25          //movimento recebe 0 quando estiver parado
26          // 0 - 1 ------ 0.1 0.2 0.3 0.5
27
28          rig.velocity = new Vector2(movimento * velocidade, rig.velocity.y);
29
30          //Altera a velocidade do componente de física do personagem para:
31          //um x que é controlado por movimento vezes velocidade e um y que
32          //apenas se repete vamos alterar y apenas no pulo
33      }
34  }
```

Fonte: Acervo próprio, 2020

A figura 7 apresenta 2 linhas de código, na primeira é realizada a captura de entrada para receber os comandos do jogador através do InputManager da Unity que pode ser acessado pelo caminho Edit>Project Settings> Input Manager, sua janela é apresentada na figura 8.

Figura 8: Captura de tela do Input Manager da Unity



Fonte: Acervo próprio, 2020

Essa funcionalidade traz a implementação de comandos comuns aos utilizados na grande maioria dos jogos, quando o eixo horizontal do input é chamado, ele passa a receber uma informação que varia de -1 a 1, caso o botão negativo seja apertado ou caso o botão positivo seja apertado respectivamente. Nessa janela é possível alterar as teclas, mas o padrão é seta direcional esquerda para negativo e seta direcional direita como positivo, sendo botões paralelos com o mesmo comportamento temos as teclas A e D.

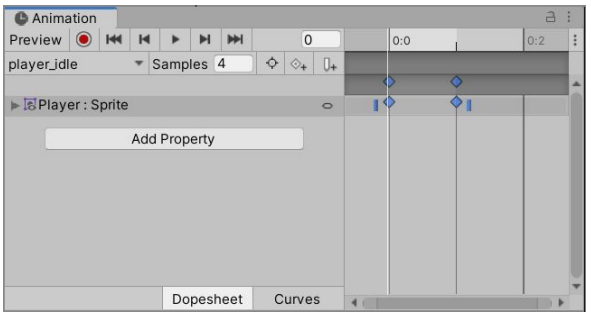
Uma alternativa similar é implementar as teclas diretamente no código, porém, utilizando o Input Manager conseguimos aplicar várias teclas com um mesmo comando, além disso ela também está pronta para receber ações do mouse e se possuir um Joystick ele também é capaz de reconhecer os botões. A segunda linha de código é chamada de velocidade do componente de física e passa um vetor composto por X e Y, dessa maneira, aplicamos em X o fator de movimento capturado do eixo Horizontal do InputManager (variação de -1 e 1) e multiplicamos pela variável de velocidade, permitindo que ele se mova para direita e para a esquerda. Já o Y, é apenas referenciado para que sua velocidade seja conservada. Os conceitos envolvendo X e Y foram apresentados de maneira prática dentro da ferramenta.

3.2.3 Terceiro encontro

Na sequência do encontro anterior, é possível mover o personagem para a direita e para a esquerda sobre o chão ou plataformas, entretanto, naquele momento ainda não está completo, o passo seguinte é adicionar as animações. Ao se movimentar o personagem deve entrar em animação de corrida, ao parar ele deve fazer uma animação simples que o mantenha se movimentando levemente, ao saltar ele deve parar qualquer outra coisa e ativar sua arte de pulo.

Para desenvolver cada animação, são utilizados alguns quadros em diferentes posições, que ao serem juntos dão a sensação de movimento, os alunos recebem os quadros separados em pastas prontas para serem inseridos na janela da Unity, na figura 9 é apresentada a Animation, área responsável pela criação das animações.

Figura 9: Captura de tela de Animation da Unity



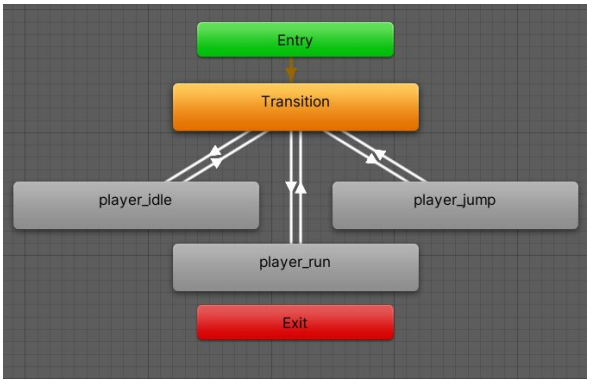
Fonte: Acervo próprio, 2020

Como é visível na figura 9, a tela de

Animation é dividida em duas partes, uma que controla a visualização, contando com play e número de amostras (Samples) de quadros por segundo, e outra com a linha de tempo onde cada ícone azul é um quadro adicionado. Essa etapa é feita apenas clicando e arrastando os quadros desejados dentro da pasta do projeto para a linha do tempo e controlando o número de Samples para adequar a velocidade de animação. A visualização é vista diretamente no personagem na janela de Scene. As animações montadas foram idle, run e jump, que respectivamente, são as animações do personagem parado, em corrida e saltando.

O processo para ativar essas animações em jogo é extremamente interessante, utiliza-se a implementação de máquina de estados, amplamente utilizado para controle de estados em circuitos elétricos e códigos diversos, um exemplo cotidiano é desenvolvido em um momento teórico antes de partir para a montagem do mesmo. A figura 10 apresenta a máquina de estados criada para o jogador na janela de Animator da Unity, responsável pelo controle de estados.

Figura 10: Captura de tela da máquina de estados do player (jogador)



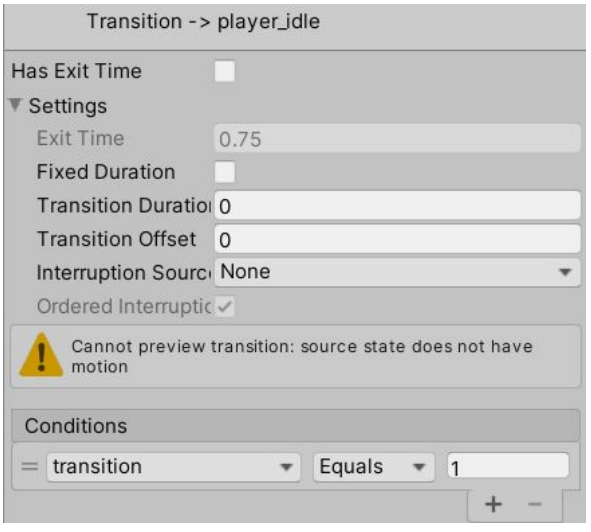
Fonte: Acervo próprio, 2020

Os blocos cinzas são os estados, o bloco laranja é o estado inicial, no bloco verde temos a chamada que ativa o estado inicial e o bloco vermelho uma saída caso seja necessário encerrar completamente a máquina. Existem inúmeras formas de se fazer o controle de estados, neste exemplo, é criado um estado vazio chamado Transition, ele é quem controla todas as transições de estado, para que cada animação consiga ser chamada é criada uma variável auxiliar que vai definir as regras de transição contidas em cada seta branca adicionada.

Dessa maneira, é cadastrada em cada flecha de transição uma regra ou um tempo de saída, como as animações não dependem de tempo e sim dos botões apertados, não se aplica regras temporais, a condição é feita a partir da troca de valor da variável auxiliar. Quando ela for igual a 1 Transition avança para player_idle, quando ela for diferente de 1 player_idle retorna para Transition, para os outros estados são utilizados os números 2 e 3, assim basta chamar no código a função que acessa essa auxiliar e alterar seu valor para mudar os estados da máquina em tempo de execução de jogo.

A variável e o controle de condições são registrados diretamente na interface da Unity de modo a facilitar esse trabalho, veja na figura 11 um exemplo de condição registrada.

Figura 11: Captura de tela da regra de transição entre Transition e player_idle



Fonte: Acervo próprio, 2020

Máquina de estados concluída, para seguir com o controle dela são feitos os seguintes passos, conforme as figuras 12 e 13.

Figura 12: Captura de tela do primeiro passo da adição de animações via script

```
12      Animator anim;
13      //variável para controlar nossas animações
14      void Start()
15      {
16          rig = GetComponent<Rigidbody2D>();
17          //Guardamos o componente de física
18          //do personagem em nossa variável
19
20          anim = GetComponent<Animator>();
21          //Guardamos o componente de animação
22          //do personagem em nossa variável
23      }
```

Fonte: Acervo próprio, 2020

Na linha 11 é criada a variável "anim" do tipo Animator, variável capaz de carregar a referência para uma máquina de estados descrita na componente Animator. Na linha 19 é utilizada a função de GetComponent para buscar o Animator registrado no objeto do jogador para a variável.

Figura 13: Captura de tela do segundo passo da adição de animações via script

```
55      if (movimento > 0)//direita
56      {
57          //aqui vai o código
58          anim.SetInteger("transition", 2);
59          //ativando a animação de corrida ao alterar
60          //transition para 2
61          transform.eulerAngles = new Vector2(0, 0);
62          //mudando os angulos de rotação do personagem
63          //através do transform
64      }
65      if (movimento < 0)//esquerda
66      {
67          //aqui vai o código
68          anim.SetInteger("transition", 2);
69          //ativando a animação de corrida ao alterar
70          //transition para 2
71          transform.eulerAngles = new Vector2(0, 180);
72          //mudando os angulos de rotação do personagem
73          //através do transform
74      }
75      if (movimento == 0)//parado
76      {
77          //aqui vai o código
78          anim.SetInteger("transition", 1);
79          //ativando a animação de idle ao alterar
80          //transition para 1
81      }
```

Fonte: Acervo próprio, 2020

Neste momento é introduzido o conceito de 'se então', é necessário avaliar a variável de movimento que como apresentado, varia de -1 a 1, conforme é possível ver na figura 13, se o movimento está acima de 0, ou seja, ao deixar 0 em direção a 1 ele está se movendo para a direita, a linha 41 faz o chamado para a variável anim do Animator e altera a variável auxiliar da máquina de estados chamada aqui de "transition", isso faz a transição disparar para o estado de corrida, a mesma lógica é utilizada quando o personagem se move para esquerda.

O movimento exatamente igual a 0 significa que está parado, então é feita a chamada na linha 61 para acionar a máquina de estados e ativar a animação de parado do jogador. As linhas 44 e 54 chamam a componente angular do personagem, ao ir para a esquerda ele precisa ser girado em 180° no eixo Y para ficar espelhado e não andar de costas, quando ele vai para a direita esse feito é revertido voltando o ângulo para 0.

A finalização deste encontro é feita com a proposta de atividade extra, montagem de um inimigo com comportamentos físicos aplicados pelos componentes da ferramenta utilizados no jogador, animações e sua máquina de estado.

3.2.4 Quarto encontro

Apresentando o código atual para os alunos, discute se todos concordam que a grande maioria das funcionalidades do jogador, em especial todas que aguardam comandos, são desenvolvidas dentro da função de Update, uma vez que ela é acionada a cada instante do jogo.
Entendendo então que as funções de movimento, pulo e outras interações vão ser dispostas neste módulo, é feita a organização do código em funções separadas, de modo que os alunos percebam sua importância na limpeza do script e reutilização de código.

Para que o jogador esteja apto a explorar todo o cenário produzido pelos jovens programadores, é preciso a capacidade de pular e que a câmera de jogo acompanhe em sua caminhada. O personagem deve ser capaz de efetuar um pulo se tiver um chão sob seus pés para que faça o impulso, pensando nisso, é adicionado um componente vazio ao pé do objeto do jogador, para que este verifique se o personagem está no chão e permita a execução do pulo.

O objeto que carrega chão e plataformas é

identificado usando a técnica de layers (camadas), a ferramenta permite categorizar objetos dessa forma, permitindo assim que seja estabelecido o controle desejado. As figuras 14, 15 e 16 demonstram o código desenvolvido em conjunto com os estudantes.

Figura 14: Captura de tela da atualização das variáveis no script do jogador para implementação do pulo

```
7       public float velocidade;
8       //esta variável é responsável
9       //pela velocidade do jogador
10      public float forcaPulo;
11      //esta variável é responsável
12      //pela força de pulo do jogador
13      bool puloDuplo;
14      //esta variável é para saber se o
15      //jogador pode realizar um pulo duplo
16      Rigidbody2D rig;
17      //Variável que vai carregar o
18      //componente de física do personagem
19      Animator anim;
20      //variável para controlar nossas animações
21      bool noTerreno;
22      //variável para saber se o personagem
23      //está no terreno ou não
24      public LayerMask terrenoLayer;
25      //variável para receber a layer do chão
26      public Transform testaTerreno;
27      //variável de referência do objeto que
28      //colocamos no pé do personagem
```

Fonte: Acervo próprio, 2020

Figura 15: Captura de tela da função de pulo no script do jogador

```
89  void Pulo()
90  {
91      noTerreno = Physics2D.OverlapCircle(testaTerreno.position,
92          0.05f, terrenoLayer);
93      //noTerreno receber verdadeiro se o circulo criado na posição do objeto
94      //no pé do personagem estiver colidindo com um objeto do layer chão
95      if(noTerreno == false)
96      {
97          anim.SetInteger("transition", 3);
98      }
99      if(Input.GetButtonDown("Jump"))
100     {
101         if (noTerreno)
102         {
103             rig.AddForce(new Vector2(0, forcaPulo),
104                 ForceMode2D.Impulse);
105             //adicionamos uma força de impulso em Y para realizar o pulo
106             puloDuplo = true;
107         }
108         else
109         {
110             if (puloDuplo)
111             {
112                 rig.velocity = new Vector2(0, 1 * forcaPulo);
113                 //adicionamos uma força de impulso em Y para realizar o pulo
114                 puloDuplo = false;
115             }
116         }
117     }
```

Fonte: Acervo próprio, 2020

Figura 16: Captura de tela da atualização da função Update no script do jogador para implementação do pulo

```
41      void Update()
42      {
43          Movimento();//Chamada da movimento no update
44          Pulo();//chamada do pulo no update
45      }
```

Fonte: Acervo próprio, 2020

O código conta também com o desenvolvimento de um segundo pulo no ar, é comum para muitos jogos do gênero e sua implementação trabalha bem a lógica das condicionais. Para o código de física que ativa o pulo, é utilizado uma força de impulso no eixo Y, de modo que o pulo tenha uma aparência natural ao lançar o jogador para cima e deixa a gravidade fazer o resto. O pulo no ar só é realizado uma vez, carregando novamente os dois pulos ao tocar o chão. É necessário que os alunos utilizem impulso no pulo normal e no segundo pulo, diferente da figura 15 que usa velocidade para o mesmo. Dessa maneira é notável que o segundo impulso apresenta um comportamento diferente, criando um salto super alto se feito rápido e criando um salto muito pequeno se demorar para ativar, é solicitado que os alunos pesquisem e discutam sobre esse fenômeno. Desse modo, adquire-se um forte entendimento sobre as forças, ao concluir que os impulsos se juntam ganhando uma elevada altitude, e quando o segundo é chamado de maneira tardia, a força de aceleração da gravidade atua sobre o personagem se opondo ao impulso.

A sequência do encontro agora é dedicada ao script da câmera principal, que conforme o código segue o personagem com um leve atraso para garantir uma transição suave e agradável. Cria-se agora o script, mas antes de escrever, é adicionada ao jogador uma etiqueta (Tag) “Player” através da janela do Inspector, dessa forma, é só orientar a posição buscando por um objeto com a etiqueta com este nome. Na figura 17 pode-se observar o script.

Figura 17: Captura de tela do script da câmera do jogo

```
Transform jogadorPos;
//variável da posição do jogador
// Start is called before the first frame update
0 references
void Start()
{
    jogadorPos = GameObject.FindGameObjectWithTag("Player").transform;
    //encontra o objeto com o tag de Player e pega sua transform
}

// LateUpdate é chamado uma vez por frame no final do frame
0 references
void LateUpdate()
{
    Vector3 pos = transform.position;
    //recebe a posição da câmera

    pos.x = jogadorPos.transform.position.x;
    pos.y = jogadorPos.transform.position.y + 2;
    //recebendo as posições x e y do player

    transform.position = Vector3.Lerp(transform.position, pos, Time.deltaTime * 3);
    //utilizando a função Lerp que faz uma mudança de posição sutil
    //recebendo uma posição inicial, uma posição final e uma varição de tempo
}
}
```

Fonte: Acervo próprio, 2020

A lógica de desenvolvimento é baseada em buscar a componente de posição do jogador, e passar suas coordenadas X e Y para um vetor auxiliar, então chama-se a função Lerp, que recebe como parâmetros uma posição inicial, uma posição alvo e uma variação de tempo, criando uma transição suave entre os dois pontos. O argumento de tempo é controlado com o Time.DeltaTime, uma componente que carrega a variação de tempo entre cada quadro de jogo.

3.3 Avaliações de desempenho dos discentes

O desempenho, acompanhamento, desenvolvimento e satisfação com a experiência é coletado a partir de diferentes fontes.

3.3.1 Gameficação da experiência

Os fatores de presença, engajamento, atividades extras e aprendizado norteiam informações valiosas sobre o acompanhamento do aluno, estas, são listadas, pontuadas e apresentadas aos alunos em uma tabela de modo a 'gameficar' a experiência através da busca pela maior pontuação no ranking final.

3.3.2 Questionário de conceitos iniciais

Os estudantes são submetidos no início do minicurso a um pequeno questionário online totalmente discursivo, a fim de medir os conhecimentos teóricos que já possuem e realizar um comparativo de desenvolvimento comparando com os demais testes. As questões avaliativas são:

Pergunta 1: Defina com as suas palavras, o que você entende por variável pública e variável privada ?
Pergunta 2: O que você entende sobre funções em programação ? Faça uma definição e conte 2 vantagens de utilizar.
Pergunta 3: O que são os argumentos(ou parâmetros) de uma função ?
Pergunta 4: O que são as bibliotecas dentro do contexto de código de programação ?

3.3.3 Questionário de avaliação de progresso

O questionário inicial é retomado com as perguntas anteriores e as novas dentro dos conhecimentos trabalhados, este verifica a mudança de discurso dentro das questões já respondidas e o desenvolvimento dos novos conhecimentos. As questões aplicadas neste teste são:

Pergunta 1: Defina com as suas palavras, o que você entende por variável pública e variável privada ?
Pergunta 2: O que você entende sobre funções em programação ? Faça uma definição e conte 2 vantagens de utilizar.
Pergunta 3: O que são os argumentos(ou parâmetros) de uma função ?
Pergunta 4: O que são as bibliotecas dentro do contexto de código de programação ?
Pergunta 5: Explique com suas palavras o que é o RigidBody2D.
Pergunta 6: O que é um Vector2 ou Vetor2D e onde utilizamos ?
Pergunta 7: Qual a diferença entre usar rig.velocity ou rig.AddForce Impulse ?
Pergunta 8: Explique o que você entendeu sobre a máquina de estados e qual a razão de utilizar em nosso jogo.
Pergunta 9: Conte sobre como foi sua experiência com os componentes da Unity.

3.3.4 Questionário de satisfação dos participantes

Os participantes devem se expressar livremente a fim de desenvolver uma pesquisa de satisfação efetiva, para tal, é criado e aplicado um questionário na plataforma Google Forms com as seguintes perguntas:

Pergunta 1: Comente um pouco sobre a diferença entre as experiências que teve com os conceitos estudados quando viram em Python vs quando viram em Unity (com c#).
Pergunta 2: O estudo através do desenvolvimento de um jogo te deixou mais motivado a aprender?
Pergunta 3: Você acredita que para o seu desenvolvimento, é mais interessante ver os conceitos separadamente em seus devidos exemplos únicos, ou conhecê-los na prática de acordo com a necessidade de utilização em um projeto ?
Pergunta 4: Acredita que o fator de distância da aula online tenha sido prejudicial ou benéfico ? Comente.
Pergunta 5: Neste último campo fale abertamente sobre a experiência, pode escrever livremente a respeito de qualquer tópico ou fazer uma consideração final.

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

4.1 Resultados observados

4.1.1 Primeiro encontro

O primeiro contato norteou os alunos através de discussão e utilização da ferramenta para composição de suas ideias. Como resultado eles se mostraram muito empolgados e engajados. Esse momento teve forte importância em aproximar a atividade com as ideias dos mesmos, segundo Grotta e Prado (2018) O estudo orientado a projetos deve estar em proximidade com a vida do aluno. Os participantes realizaram com ímpeto a atividade de desenho de mapa do jogo na Unity seguindo suas ideias individuais.

4.1.2 Segundo encontro

A segunda aula cumpriu satisfatoriamente seu papel, os estudantes puderam ver os comportamentos de física dos componentes acontecendo em tempo real. Como afirmado por Comber et al (2019) Há uma redução de complexidade por parte da Unity devido ao fornecimento de componentes de física. Experimentalmente os alunos diferenciavam as características para o corpo do personagem e para o cenário com o qual interage, ajustando suas regras e testando até atingir o objetivo desejado de que o personagem consiga se manter em pé nas plataformas. A introdução ao script ainda não parecia firme enquanto abordada no universo teórico, passou a tomar forma diante os resultados visuais.

4.1.3 Terceiro encontro

O terceiro momento do minicurso trouxe uma introdução a máquinas de estado, todos os alunos demonstraram compreensão quando desenvolvida em conjunto, e apenas dois dos cinco confirmaram ter dominado o assunto através da atividade extra de criação de um inimigo e o controle de suas animações via máquina de estados. Um aluno não apresentou esse desafio por não estar presente na aula final devido uma demanda de provas da sua escola regular. Um outro aluno também não apresentou devido a problemas em sua máquina. Apesar disso, conhecer sobre esse assunto e suas aplicações gerou interesse nos alunos assim como observado por Marques (2011) É visto que a curiosidade dos alunos pela área de informática é aumentada na utilização de jogos como ferramenta de instigação.

4.1.4 Quarto encontro

Um dos principais resultados do encontro final foi o trabalho de lógica, lidando com mais severidade em condições e variáveis de verdadeiro ou falso, juntamente com o estudo dos efeitos da força de impulso no pulo versus gravidade na solução dos problemas. Os alunos se empolgaram com as discussões e tentativas de correções,

atestando conformidade com o que afirma Carosia et al (2017) Desenvolver um jogo é algo estimulante, e provoca a solução de problemas. Três dos estudantes apresentaram o estado final do desenvolvimento com sucesso, evidenciando que foram capazes de avançar diante todas as etapas. Os outros dois estudantes não conseguiram apresentar devido a indisponibilidade neste encontro, porém demonstraram completude nos conteúdos trabalhados nos encontros anteriores, dessa maneira, os alunos demonstraram efetivos resultados práticos.

4.2 Resultados coletados via questionário e dados gerados

Essa seção traz a síntese dos dados avaliativos definidos para cada um dos estudantes, e as informações que podem ser extraídas a partir dos mesmos.

4.2.1 Resultado questionário de avaliação inicial

Os discursos apresentados pelos alunos foram avaliados e pontuados de 0 a 1, permitindo um acúmulo de pontuação máximo de 4 pontos. O questionário aproveita qualquer linha de raciocínio coerente com os conceitos. As pontuações de cada um estão dispostas na tabela 1 abaixo.

Tabela 1. Pontuações dos alunos no questionário inicial

Aluno	1	2	3	4	5
Q1	0	1	0	1	1
Q2	0.7	0.7	0.2	0.4	0.8
Q3	0.5	1	0.4	0.2	0.5
Q4	1	1	0.2	0	1
Total	2.2	3.7	0.8	1.6	3.3
% de Acerto	55%	92.5%	20%	40%	82.5%

Fonte: Acervo próprio, 2020

A tabela 1 ilustra uma variedade de domínio dos conceitos, mas registra essas informações para a comparação com as demais análises.

4.2.2 Resultado questionário de avaliação de progresso

A tabela 2 contempla as primeiras quatro perguntas do teste inicial somadas a mais quatro questões de conhecimentos adquiridos durante o minicurso.

Tabela 2. Pontuações dos alunos no questionário final

Aluno	1	2	3	4	5
Q1	0.8	1	0.5	1	1
Q2	0.9	0.9	0.6	0.5	0.8
Q3	0.5	1	0.4	0.4	0.5
Q4	1	1	0.2	0.7	1
Parcial	3.2	3.9	1.7	2.6	3.3
% de Acerto	80%	97.5%	42,5 %	65%	82.5 %
Q5	1	1	0.5	1	1
Q6	0.9	0.8	0.5	0.8	0.8
Q7	0.8	0.8	0.6	0	0.8
Q8	0.8	0.7	0.7	0.5	0.8
Total	6.7	7.2	4	4.9	6.7
% de Acerto	83,7 5%	90%	50%	61,25 %	83,7 5%

Fonte: Acervo próprio, 2020

A tabela 2 apresenta as porcentagens finais dos conceitos teóricos base somando os conteúdos teóricos de programação com a Unity para uma totalidade de aproveitamento de conceitos. Correlacionando os resultados, é demonstrada uma evolução argumentativa dentro das teorias para todos os participantes exceto o número 5 que se manteve constante em seus discursos. Notando um aumento de assertividade nas questões Q1 a Q4 que trabalhavam os mesmos conceitos do primeiro teste. Os alunos 3 e 4 não aproveitaram a prática do quarto encontro, este teve um trabalho muito focado nos códigos finais, portanto, é considerado uma provável razão para notas menores nas quatro últimas questões e uma evolução menos impactante dentro dos conceitos iniciais.

4.2.3 Resultado Gameficação

Através da gamificação citada em 3.3.1, e análise a partir dos formulários foram obtidos os resultados da tabela 3

considerando 19 a pontuação máxima.

Tabela 3. Pontuações dos alunos em cada fator avaliado e total

Aluno	1	2	3	4	5
Presença	4	4	4	3	4
Engajamento	4	4	4	4	4
Aprendizado	6.7	7.2	4	4.9	6.7
Extras	2	3	2	2	3
Total	16.7	18.2	14	13.9	17.7
% de Aproveitamento	87.9 %	95.8 %	73.7%	73.15 %	93.15 %

Fonte: Acervo próprio, 2020

Presença e engajamento tem notas máximas em 4, delimitadas dentro do número de encontros. Aprendizado tem nota máxima 8 e é extraído da tabela de pontuação em conceitos. Os pontos extras são compostos por um total de 3 atividades sugeridas durante os encontros. Diantes os resultados da tabela 3, os alunos apresentaram ótimas porcentagens de aproveitamento com exceção dos estudantes 3 e 4, que estão dentro do satisfatório levando em conta a não realização da atividade do encontro final.

4.2.4 Resultado questionário de satisfação

Pergunta 1: Comente um pouco sobre a diferença entre as experiências que teve com os conceitos estudados quando viram em Python vs quando viram em Unity(com c#).

A resultante dos discursos dos alunos ilustra nesta questão que Python mostrava uma simplicidade em escrever os códigos mais básicos, mas uma maior dificuldade quando tentaram aplicar aquele conhecimento em desenvolvimento de jogos.

Em Unity destacaram que era mais fácil atingir objetivos práticos com menos código, pois a ferramenta auxilia bastante com seus componentes e janelas de apoio ao desenvolvimento, permitindo um código mais focado e objetivo.

Pergunta 2: O estudo através do desenvolvimento de um jogo te deixou mais motivado a aprender?

Todos os 5 alunos responderam de maneira afirmativa, confirmando a motivação apresentada por eles nos encontros.

Pergunta 3: Você acredita que para o seu desenvolvimento, é mais interessante ver os conceitos separadamente em seus devidos exemplos únicos, ou conhecê-los na prática de acordo com a necessidade de utilização em um projeto ?

De forma geral, os alunos concluíram que conseguem se adaptar melhor aos conhecimentos acompanhando na prática em um projeto real, destaca-se a resposta do aluno número 4, que afirmou que esta metodologia estimula a mente a lidar com maneiras de resolver os problemas encontrados durante o desenvolvimento.

Pergunta 4: Acredita que o fator de distância da aula online tenha sido prejudicial ou benéfico ? Comente.

Embora essa pergunta seja muito particular, os fatores prejudiciais apresentados foram apenas problemas com internet ou com a máquina pessoal. Os alunos que destacaram essas dificuldades, complementam que as gravações das aulas os permitiram rever e lidar com mais calma com os conteúdos abordados.

Pergunta 5: Neste último campo fale abertamente sobre a experiência. Pode escrever livremente a respeito de qualquer tópico ou fazer uma consideração final.

Nesse discurso os alunos falaram sobre o gosto por programação que possuíam e destacaram um interesse maior ainda após as aulas. Alguns demonstraram sentir uma forte importância do curso para seu futuro, um trecho a ser destacado dos demais foi o do aluno 2 “gostei bastante, aprendi muito e já me vejo fazendo um jogo, o conteúdo foi suave e bem explicado, gostei muito e quero aprender mais” em suas palavras,

confirmando sua satisfação.

5 CONSIDERAÇÕES FINAIS

A metodologia atestou sucesso para com a abordagem dos conteúdos trabalhados, confirmando ter uma aplicação engajadora e motivacional como proposto, entregando conteúdo prático e demonstrando evoluções em conceitos teóricos. O fator de tempo não permitiu que mais encontros fossem realizados, dessa maneira os alunos não foram capazes de atingir a completude de seus projetos, porém eles se mostram ainda engajados após o encontro final, solicitando as gravações e buscando implementar novas funcionalidades, o que afirma ainda mais o fator motivacional objetivado.

O papel da Unity como ferramenta auxiliar foi extremamente importante para a experiência prática, trazendo demonstrações dentro dos testes do jogo de modo visual, sendo base para o pensamento lógico das soluções de problemas.

Mais estudos devem ser feitos na área de pesquisa, em especial estudos que detalham os conteúdos trabalhados em aula, que são de suma importância para a reprodução e melhoria das técnicas de ensino.

REFERÊNCIAS

ANDRADE, José Raul B. Et al. **Uma Proposta de Oficina de Desenvolvimento de Jogos Digitais para Ensino de Programação**. Anais dos Workshops do V Congresso Brasileiro de Informática na Educação (CBIE), 2016.

COUTINHO, Emanuel F et al. Relato sobre **o Uso de uma Ferramenta de Desenvolvimento de Jogos para o Ensino Introdutório de Lógica de Programação**. Anais dos Workshops do VII Congresso Brasileiro de Informática na Educação (WCBIE). 2018.

CAROSIA, Arthur Emanuel de Oliveira et al. **3dponk: O Desenvolvimento De Jogos Para Ensinar Linguagem De Programação No Ensino Médio**. Revista Científica do Instituto Federal de São Paulo. 2018.

CARVALHO, Leandro S. G et al. **Juiz online como ferramenta de apoio a uma metodologia de ensino híbrido em programação**. Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE), 2016.

COMBER, Oswald et al. **Engaging Students in Computer Science Education through Game Development with Unity**. IEEE Global Engineering Education Conference (EDUCON).2019.

FALCÃO, Emerson Uriel e JÚNIOR, José Luiz Ungeritch. **Desenvolvimento de jogos eletrônicos como metodologia de ensino de programação para alunos do curso de informática do instituto federal catarinense – câmpus Camboriú.** Mostra Nacional de Iniciação Científica e Tecnológica Interdisciplinar. Instituto Federal de Santa Catarina, 2015.

GOMES, Tancicleide C. S. Et al. **Jogos no design de experiências engajadoras**. V Congresso Brasileiro de Informática na Educação(CBIE). 2016.

GROTTA, Alexandre e PRADO, Edmir P.V. **Um ensaio sobre a experiência educacional na programação de computadores: a abordagem tradicional versus a aprendizagem baseada em projetos.** Anais do XXVI Workshop sobre educação em computação. São Paulo, 2018.

HOED, Raphael Magalhães. **Análise da evasão em cursos superiores: o caso da evasão em cursos superiores da área de Computação.** Dissertação (Mestrado Profissional em Computação Aplicada) —Universidade de Brasília, Brasília, 2016.

MARQUES, Diego Lopes. Et al. **Atraindo alunos do ensino médio para a computação: uma experiência prática de introdução e programação utilizando jogos e python.** Anais do XXII de Simpósio Brasileiro de Informática na Educação. Aracaju, 2011.

PAULA, Leda Queiroz. Et al. **A Importância da Leitura e da Abstração do Problema no processo de formação do raciocínio lógico-abstrato em alunos de Computação**. Revista de Estudos e Reflexões Tecnológicas da Faculdade de Indaiatuba, 2009.

UNITY. **Unity Manual**. 2020. Disponível em:<https://docs.unity3d.com/Manual/index.html>. Acesso em: 01 de novembro de 2020.

VALASKI, Joselaine e PARAISO, Emerson Cabreira. **Limitações da utilização do Alice no Ensino de Programação para Alunos de Graduação**. Anais do Simpósio Brasileiro de Informática na Educação (SBIE). 2012.