VICTOR MANOEL VILELA BRITO

DESENVOLVIMENTO DE JOGO DIGITAL PARA CONSCIENTIZAÇÃO ECOLÓGICA

TEÓFILO OTONI – MG
FACULDADES UNIFICADAS DE TEÓFILO OTONI
2016

VICTOR MANOEL VILELA BRITO

DESENVOLVIMENTO DE JOGO DIGITAL PARA CONSCIENTIZAÇÃO ECOLÓGICA

Monografia apresentada ao Curso de Sistemas de Informação das Faculdades Unificadas de Teófilo Otoni, como requisito parcial à obtenção do título de Bacharel em Sistemas de Informação.
Área de Concentração: Arte e Desenvolvimento.
Orientador: Msc. Salim Ziad Pereira Aouar.

TEÓFILO OTONI – MG
FACULDADES UNIFICADAS DE TEÓFILO OTONI
2016

rede de ensino
DOCTUM

FACULDADES UNIFICADAS DE TEÓFILO OTONI
NÚCLEO DE TCC / SISTEMAS DE INFORMAÇÃO
Autorizado pela Portaria 4.012 de 06/123/2004 – MEC

FOLHA DE APROVAÇÃO

A monografia intitulada: *Desenvolvimento de jogo digital para conscientização ecológica,*

elaborada pelo aluno Victor Manoel Vilela Brito,

foi aprovada por todos os membros da Banca Examinadora e aceita pelo curso de Sistemas de Informação das Faculdades Unificadas de Teófilo Otoni, como requisito parcial da obtenção do título de

BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Teófilo Otoni, 22 de novembro de 2016

Professor Orientador: Salim Ziad Pereira Aouar

Professor Examinador: Amaury Gonçalves Costa

Professor Examinador: Luiz Fernando Alves

Dedico este trabalho, aos meus pais,
especialmente minha irmã Helen Vilela Brito,
que desenhou todos os sprites para a
realização deste projeto.

AGRADECIMENTOS

Aos meus pais, familiares e especialmente minha irmã; ao Patrick (professor das video-aulas do youtube); ao Carlos W. Gama (autor dos tutoriais do site jogosindie.com); aos meus amigos Andre Ricardo e José Gomes; ao Prof. Salim Ziad Pereira Aouar; e a todos que apoiaram a realização deste trabalho. A todos os professores do curso, pela dedicação que os mesmos disponibilizaram nas aulas; ao pessoal da Unity Technologies, e a Deus, que foi quem mais ajudou e esteve sempre ao meu lado.

“Para mudar, não basta apenas querer, é preciso ter coragem.”

RESUMO

O presente trabalho tem como principal objetivo desenvolver um jogo eletrônico seguindo o conceito de conscientização ecológica e utilização da tecnologia Unity. Trata-se de um desenvolvimento na área de jogos digitais cujo título seja “Desenvolvimento de jogo digital para conscientização ecológica”. Serão apresentadas as ferramentas e as metodologias utilizadas neste projeto. A principal metodologia utilizada foi o uso da ferramenta Unity, destacando suas funções para o desenvolvimento dos códigos, animações, física e diversas outras funcionalidades utilizadas no jogo. Em seguida, serão demonstradas as técnicas utilizadas na criação dos sprites dos personagens, objetos, animações, entre outros. Por fim, a utilização em conjunto das tecnologias Unity, Paint tool SAI trazem resultados muito eficazes, resultando em um jogo fácil de programar e implementar mais conteúdo, aumentando ainda mais a sua qualidade.

Palavras-chave: Unity; desenvolvimento; conscientização ecológica; sprites; jogo eletrônico.

ABSTRACT

The present essay has as main objective to develop an electronic game following the concept of ecological awareness and use of Unity technology. It is about a development in the area of digital games whose title is "Development of digital game for ecological awareness". The tools and methodologies used in this project will be presented. The main methodology used was the Unity tool, highlighting its functions for the development of codes, animations, physics and several other functionalities present in the game. Then, the techniques used to create the sprites of the characters, objects, animations, among others will be demonstrated. Finally, the combined use of Unity, Paint tool SAI technologies bring very effective results, coming out as an easy game to program and implement more content, further enhancing quality.

Key-words: Unity; development; ecological awareness; sprites; electronic game.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

INTRODUÇÃO

Trata-se de um desenvolvimento na área de jogos digitais cujo título é “Desenvolvimento de jogo digital para conscientização ecológica”. O objetivo deste projeto é desenvolver um jogo eletrônico sobre conscientização ecológica, utilizando a tecnologia Unity. Os problemas apresentados serão: o desmatamento e a poluição da água. Visa contribuir como ferramenta educativa em relação aos problemas ecológicos mediante um cenário que busca o despertar da consciência ecológica no jogador, considerando que, no mundo atual a preservação ambiental é uma preocupação crescente por parte do governo, das pessoas e das organizações.

A metodologia utilizada na execução deste projeto apresentou as seguintes etapas:

- O primeiro capítulo apresenta a teoria e as ferramentas utilizadas no desenvolvimento deste projeto. Foi utilizado o Unity para aplicação de animações, movimentações, física e outras demais funcionalidades.
- O capítulo seguinte apresenta os materiais e métodos utilizados no desenvolvimento gráfico do projeto, desde planejamento, personagens, cenários e objetos.
- Foi utilizado o dispositivo tablet gráfico e o programa Paint tool SAI para o desenvolvimento dos sprites dos personagens, cenários e itens.
- Foi utilizado o compilador monodevelop para o desenvolvimento dos scripts.
- Foi utilizado o Photoshop para concluir os desenhos dos sprites.
- Foram utilizadas pesquisas bibliográficas em livros e Internet para referenciar a teoria do projeto.

O principal objetivo deste projeto é desenvolver um jogo eletrônico seguindo o conceito de conscientização ecológica tendo como metas: desenvolver conceitos de conscientização relacionados ao desmatamento e a poluição da água.

O elemento principal deste trabalho é o jogo eletrônico. Trata-se de um software e alguns problemas serão apresentados, então foram encontradas as seguintes hipóteses:

HO: Não seria viável realizar o estudo, pois o desenvolvimento de jogo eletrônico não seria capaz de ajudar na conscientização ecológica.

H1: Para desenvolver um jogo eletrônico é necessário saber programar.

H2: É possível desenvolver um jogo eletrônico que ajude na conscientização ecológica.

Os ganhos dos estudos começaram logo no início do projeto, aplicando o conceito de conscientização ecológica e desenvolvendo o jogo com originalidade até o final do projeto.

Os capítulos deste trabalho apresentam os seguintes conteúdos:

- O primeiro capítulo: revisão literária – apresenta os capítulos teóricos do projeto, onde aborda as funcionalidades do Unity e algumas ferramentas que foram utilizadas no desenvolvimento do jogo.
- O segundo capítulo: materiais e métodos – apresenta o desenvolvimento gráfico e planejamento do projeto.
- O terceiro capítulo: sistema – apresenta a ficha técnica do jogo e uma breve explicação sobre os códigos, telas e os testes.
- O quarto capítulo: resultados e discussão – apresenta os resultados do projeto e os motivos que levaram ao mesmo.
- O quinto capítulo: considerações finais – apresenta a conclusão do projeto e as análises das validações das hipóteses.
- O sétimo capítulo: apresenta os principais códigos utilizados no projeto.

1 REVISÃO LITERÁRIA

1.1 CONCEITO

O jogo se baseia no conceito de salvar um determinado ecossistema em uma área fictícia cujo problema é o desmatamento e a poluição da água. Esses problemas serão apresentados como conteúdo do jogo para trabalhar a conscientização ecológica, já que, segundo Silva e Pozzi apud Silva (2015, p.13) “Numa visão mais diretiva e objetiva, o jogo é apresentado como uma ação que acontece diante de certos momentos, espaços e tempos, construindo e validando o desenvolvimento integral e cultual do seu respectivo jogador”.

Utilizando o tema sobre problemas ecológicos em um jogo digital e criando conteúdos relacionados a esse tema como os fatores que causam o desmatamento e a poluição, demonstrará os efeitos nocivos ao desmatar uma floresta ou poluir o oceano. Isso mostrará a importância da conservação ecológica, incentivando a conscientização e a preservação da natureza.

Segundo Freire (1970, p.15):

> A conscientização implica, pois, que ultrapassemos a esfera espontânea de apreensão da realidade, para chegarmos a uma esfera crítica na qual a realidade se dá como objeto cognoscível e na qual o homem assume uma posição epistemológica.

1.2 UNITY

Para o desenvolvimento deste projeto, foi escolhido o software Unity, que em suas licenças de uso encontra-se uma versão gratuita, já que, segundo Viana (2009, p.3) “O Unity é um programa de desenvolvimento com todos os recursos necessários para desenvolver qualquer tipo de jogo que você possa imaginar. O Unity na grande maioria das vezes é usado na criação jogos de browser, (Jogos que rodam em navegador de internet). Mas o motor é capaz de criar grandes jogos. Ele possui uma versão gratuita, que pode ser encontrada no link http://unity3d.com”.

1.3 LINGUAGEM DE PROGRAMAÇÃO

Os programas de computador (também chamado de software) são formados por um conjunto de comandos para serem executados no computador. Para criar um programa, é preciso especificar os comandos para o computador realizar as tarefas desejadas. Os métodos padronizados para especificar os comandos são chamados de linguagem de programação.

Segundo Watkins (2012, p. 334):

> O Unity usa três linguagens de script: JavaScript(UnityScript), C# e Boo. Essas três linguagens de script são maneiras muito diversas de acessar a mesma funcionalidade básica do mecanismo do jogo; três ferramentas com o mesmo objetivo.

1.4 LINGUAGEM C#

C# é uma linguagem de programação orientada a objetos desenvolvida pela Microsoft, ela é utilizada pela Unity para programar suas funcionalidades e recursos de desenvolvimento de jogos.

Segundo Lotar (2010, p.32):

> C# é uma linguagem de programação simples, porém poderosa e ao mesmo tempo ideal para desenvolver aplicações web com ASP.NET. É uma evolução do C e C++. O C# é uma linguagem orientada a objetos com a qual podemos criar classes que podem ser utilizadas por outras linguagens.

1.4.1 Monodevelop

O MonoDevelop é o compilador utilizado para desenvolver os códigos em C#, ou seja, todos os scripts de jogos criados no Unity, são construídos no MonoDevelop.

Segundo Pereira, Oliveira e Junior (2015, p. 4):

> MonoDevelop é uma IDE multi-plataforma com várias funcionalidades tais como suporte a complemento de código para C#, modelos de código e suporte a diversas linguagens de programação como C#, F# e mais. Possui integração com o Unity, o que faz dessa ferramenta muito útil no desenvolvimento de códigos para serem usados na Game Engine.

1.5 VARIÁVEIS

Variáveis são elementos utilizados em qualquer código de programação, elas podem ser do tipo numérico, alfanumérico ou lógico. Variáveis string recebem alfanuméricos, int recebe apenas números inteiros, negativos e positivos, float recebe números fracionados e bool recebe apenas valores de verdadeiro ou falso.

1.6 SCRIPTS

Os scripts funcionam como um roteiro que irá programar quando e onde deve ser feito os comandos durante o jogo. Os scripts são códigos de programação responsável por dar funcionalidade a quase tudo dentro do jogo, por isso são

necessários criá-los porque os objetos necessitam de componentes e scripts. Os componentes são como variáveis nos scrips que permitirá fazer as funcionalidades fundamentais do jogo. Sem os scripts o jogo simplesmente não funciona, por isso, a programação é essencial pra dar vida ao jogo.

1.7 CENÁRIOS

Os cenários são os Backgrounds das fases, cujo design será baseado nos conceitos de conscientização. Cada cenário será feito com base em uma situação ecológica, representando temáticas de desmatamento e poluição da água.

1.8 CÂMERA

A câmera é o componente responsável pela transmissão da cena, nela configura a altura e a largura da tela, ou seja, o cenário deve ser ajustado de acordo com as posições da câmera, para que ela possa capturar todo cenário e transmitir a cena corretamente.

1.9 MOVIMENTAÇÃO DO CENÁRIO

O cenário é ajustado para aparecer dentro da posição configurada da câmera, ou seja, o cenário deve ser do mesmo tamanho da câmera, e os objetos devem estar em posições dentro da câmera para aparecem na tela do jogo. Uma técnica chamada paralax foi utilizada para mover o cenário. A mesma funciona quando o cenário é programado para se mover junto com o sprite do personagem controlado, mas para que isso funcione corretamente, o começo do cenário deve ser ligado com o final, para dar a impressão de “mundo infinito”.

1.10 SPRITES

Sprites são desenhos criados para utilizarem em jogos 2D. Um jogo é composto por sprites de cenários, personagens ou qualquer outro objeto. Os sprites dos personagens ou quaisquer objetos são desenhados com transparência para serem colocados na frente dos cenários. Para formar os sprites dos personagens ou de algum objeto que tenha mais de uma forma, são desenhadas várias imagens para compor os movimentos. Sprites de personagens costumam ter muitos movimentos; segundo Clua e Bittencourt (2005, p.7) “Sprites: imagens bidimensionais que se movimentam umas sobre as outras criando ilusão de profundidade e controle dos resultados.”

1.11 COMPONENTES

A ferramenta Unity possui vários componentes para criação de jogos, os componentes utilizados no projeto foram para física, colisão, animação, sons e controles que são fundamentais para a composição dos gráficos e da jogabilidade.

Segundo o site da Unity (2014):

> Ao vincular Componentes a um GameObject, você cria comportamento e movimento e define a aparência. Luzes, malhas, efeitos especiais, áudio, câmeras e emissores de partículas são exemplos de Componentes. Por sua vez, cada Componente tem um conjunto próprio de propriedades ajustáveis, alcance e intensidade para luz.

1.12 GAME OBJECT

Basicamente qualquer conteúdo no projeto é chamado de objeto, entre eles existem objetos sólidos, inanimados e até seres vivos como personagens, plantas e animais. GameObjects são objetos fundamentais que representam personagens, itens e cenários, eles agem como recipiente para implementar componentes nos

objetos, ou seja, para implementar componentes nos sprites e implementar efeitos, funcionalidades e scripts.

1.13 INSPECTOR

O Inspector é uma aba de propriedades do GameObject, ela informa detalhadamente os parâmetros do objeto e permite a configuração das escalas, posições e rotações de forma matemática para cada um deles, calculando os ângulos, posições e tamanhos X, Y e Z, formando um vetor de 3 posições. A posição X configura o ângulo horizontal e a posição Y configura o ângulo vertical. A posição X ao receber um número positivo configura o objeto para a direita, se receber um número negativo configura para a esquerda. A posição Y ao receber um número positivo configura o objeto para cima, se um número negativo configura o objeto para baixo. Todos os objetos possuem essas configurações de posições. A cena do jogo é ilimitada, as posições numéricas para X e Y podem ser infinitas, ou seja, a cena pode ter qualquer tamanho. Quando for necessário modificar essas posições nos scrips, é utilizado o Vector3 que é um vetor com três posições (x,y,z). A posição z só é utilizada para jogos 3D.

1.14 HIERARCHY

A janela Hierarquia contém uma lista de todos os GameObject na cena atual. Segundo Viana (2009, p.19) “A visão da hierarquia da cena mostra todos os elementos presentes na cena, cada GameObject”. Alguns deles são instâncias diretas de Sprites, que são objetos personalizados que compõem a maior parte do jogo. À medida que os objetos são adicionados e removidos na Cena, eles aparecerão e desaparecerão da Hierarquia também. Por padrão, os objetos são listados na janela Hierarquia na ordem em que são feitos. É possível reordenar os objetos arrastando-os para cima ou para baixo ou tornando-os objetos "filho" ou "pai.

1.15 MATERIAL

O material é um recurso adicionado em um sprite de cenário do tipo textura, geralmente utilizado em alguma parte do cenário como chão, paisagem ou qualquer coisa que faça parte do cenário, como objetos flutuantes, ou qualquer objeto parado no cenário em que é possível subir em cima, como escadas, plataformas, pontes, etc. Quando um material é importado, o shader é importado automaticamente para mostrar como o material vai reagir.

Segundo Watkins (2012, p. 63):

> Texturas são imagens; elas podem ser fotos ou imagens pintadas. Texturas são elementos 2D que definem determinado atributo de um material. Texturas podem ser usadas para definir os elementos táteis, visual ou coloridos de um material, como relevo. Quando um material é aplicado a um objeto, shader é usado para mostrar como esse material vai reagir à luz e ao angulo de visão do observador.

1.16 FÍSICA

O Unity tem um conjunto de componentes para física chamada Physics, um pacote de recursos para adicionar física ao jogo, como gravidade e colisão. Esses componentes são utilizados nos objetos para dar física às coisas dentro do jogo, tornando estes componentes fundamentais para a criação do mesmo.

1.16.1 Gravidade

O componente de gravidade se chama Rigidbody, e serve para aplicar gravidade nos objetos. Esse componente é geralmente aplicado em sprites de personagens e qualquer coisa que caia no chão. Graças a este recurso os sprites dos personagens podem pular sem ficar flutuado. Rigidbody é um dos componentes de física do Unity, este recurso é essencial para a gravidade do jogo.

1.16.2 Colisão

O componente de colisão se chama Box Collider 2D. Este componente permite a colisão entre os objetos. Ao adicioná-lo, algumas funções estarão ativas. Entre elas o OnCollisionStay2D que é utilizado para dizer que um objeto está encostado no outro. Este recurso normalmente é utilizado para mostrar que um objeto colidiu com o outro, e enquanto eles estiverem colidindo pode acontecer qualquer evento programado, por isso várias coisas podem ser feitas com este recurso, tornando-o um componente essencial para a criação do jogo.

1.17 ANIMAÇÃO

A animação é a parte responsável por animar os sprites presentes no jogo. Os sprites dos personagens possuem mais de um movimento, então é utilizando a animação para organizá-los.

Segundo Watkins (2012, p. 320):

> Pense nas animações como componentes de GameObjects. Sempre que um objeto deve ser animado no Unity, o Unity cria um arquivo separado editável (com um rótulo, anim) que armazenará as informações. O que isso significa é que a animação pode ser usada repetidas vezes em outros objetos (o que é um poderoso recurso por si só). Isso também significa que há um novo recurso armazenado na pasta recursos.

1.18 ANIMADOR

O animador funciona como um fluxograma, servindo para indicar as transições dos sprites, possibilitando que os sprites possam realizar transições entre si de acordo com os comandos programados, podendo trabalhar com variáveis float, int, bool e trigger para utilizar nos scripts e criar os comandos.

1.19 JOGABILIDADE

Jogabilidade é o termo utilizado para explicar a mecânica de controle dos personagens do jogo. A jogabilidade deste projeto será simples, intuitiva, fácil de entender e terá comandos básicos como andar para direita ou esquerda, pular e atacar.

1.20 CONTROLE

O Unity possui um recurso chamado Input que é responsável por configurar os botões do jogo. Esse recurso possibilita a configuração do controle de acordo com a plataforma que o projeto pretende ser desenvolvido. Os botões possuem comandos específicos, os principais são: GetButtonDown, GetButton, GetButtonUp, GetAxis.

1.20.1 Botões

O GetButtonDown é utilizado quando é necessário ativar o comando apenas uma vez para cada vez que o comando for ativado. O GetButtonUp é utilizado quando o botão deixa de ser pressionado. O GetAxis é utilizado para comandos opostos, como direita ou esquerda, horizontal ou vertical. O GetAxis funciona quando programa os botões positivo (1) e o negativo (-1). O positivo (1) pode programar para a direita e o negativo (-1) para a esquerda.

1.21 PLATAFORMA

O Unity tem uma opção chamada build settings, onde é configurada a plataforma que o jogo será lançado. Entre as opções presentes está Windows,

Android, Windows phone, PS3, PSVita, PS4, XBOX ONE, XBOX 360 e algumas outras plataformas atuais.

1.22 SONS

Os objetos podem ter efeitos sonoros através de um componente chamado Audio Source. Ao adicionar esse componente, é possível escolher algum arquivo de áudio que contém o efeito sonoro desejado para o objeto.

Segundo Novak (2010, p. 272):

> O áudio para games pode consistir em sons sampleados (gravados), como vozes e música; sons de interface, como ruídos eletrônicos e cliques de botão; e efeitos sonoros do game, como explosões e passos. O áudio pode ser extremamente importante para a atmosfera do game, tanto para definir o clima como para alterá-lo. Pode-se usar áudio para fornecer indicações sonoras ao jogador, intensificar sua satisfação e aumentar a qualidade de um game.

1.23 TABLET GRÁFICO

O tablet gráfico é um dispositivo periférico fabricado para desenhar qualquer coisa diretamente no computador, esse dispositivo será utilizado para desenhar todo o gráfico do jogo, como cenários e sprites.

1.24 PAINT TOOL SAI

O Paint tool SAI é um software gratuito, desenvolvido pela Systemax Software Development. Este programa possui características de ser leve e de alta qualidade para desenhos e pinturas, com incrível suporte para arte digital, podendo fazer surpreendentes pinturas anti-aliasing, proporciona uma operação fácil e estável, o que faz com que nesse programa a arte digital seja mais agradável e confortável.

Essa ferramenta será utilizada na criação dos sprites desenhados na mesa digitalizadora ou tablet gráfico. A função mais utilizada são as layers, sistema que utiliza uma ou várias camadas divisórias que auxiliam e otimizam o processo desde o esboço até a pintura finalizada. Uma das maiores características do PTS é o fato dele ser anti-aliasing, ou seja, anti-serrilhamento, que serve para deixar a imagem limpa e natural.

Segundo E. Morimoto (2007, p. 812):

> Os algoritmos de antialiasing são chamados genericamente de "FSAA" (Full-Screen Antialiasing). A ideia básica é suavizar as imagens (sobretudo os contornos), reduzindo a granulação e tornando a imagem mais "lisa", de forma que ela aparente uma resolução maior que a real.

1.25 PHOTOSHOP

O Photoshop é um software para edição de imagens. Neste projeto, será utilizado para aplicar transparência no fundo dos sprites, ou seja, a transparência é a etapa final na construção dos sprites, sendo necessária para possibilitar a montagem dos personagens com o cenário. Segundo Furlan (2012, p.4) “O Photoshop não é apenas uma aplicação de edição de imagens qualquer, é a mais avançada e mais abrangente aplicação de edição de imagem profissional.”

1.26 ESTILO MANGÁ

Os sprites são desenhados em estilo mangá, que é um estilo de desenho tipicamente japonês.

Segundo Vincent (2016):

> Além dos exageros ao representar emoções e ações também é típico das histórias em mangás. Olhos muito arregalados para demonstrar expressões de surpresa, páginas com enormes onomatopéias, desenhos que ocupam uma página inteira quando um personagem faz uma demonstração de seu poder; o exagero faz parte desta forma de expressão.

De acordo com o site www.resumoescolar.com.br, até mesmo os traços pelos quais o mangá ficou conhecido, olhos muito grandes, cabeça desproporcional ao corpo, pernas longas, são um reflexo do seu exagero característico. Durante o processo de criação dos desenhos são criados vários esboços até chegar a uma ideia do que será feito. Logo após, é feito um esboço mais claro da versão escolhida, assim, faz-se a arte final.

2 MATERIAIS E MÉTODOS

Os cenários utilizados no jogo serão conceituados, cada cenário irá conter um tema referente a um problema ecológico e um objetivo a ser cumprido durante o jogo. O desmatamento será um tema utilizado na fase 1, que vai ter como objetivo parar o desmatamento ilegal. A poluição da água é um tema que será utilizado na fase 2, cujo objetivo é parar a fonte de resíduos jogados ilegalmente na água.

2.1 UNIVERSO DO JOGO

O projeto é desenvolver um jogo do gênero plataforma, com o objetivo de salvar uma floresta e um oceano de um desastre ecológico. O jogador terá que explorar todo o cenário e deter todos os agentes responsáveis pelo desmatamento ilegal e pela fonte de resíduos jogados no oceano.

2.2 DESENHO DOS CENÁRIOS

Desenhar os cenários foi uma das etapas mais importantes deste projeto, para desenhar os cenários foi utilizado o Tablet gráfico e o Paint tool SAI, por serem ferramentas eficientes e com muitos recursos.

Figura 1: Desenho do cenário: Floresta.

Fonte: Elaborada pelo autor

Figura 2: Desenho do cenário: Praia.

Fonte: Elaborada pelo autor.

2.3 CRIANDO UM PROTÓTIPO

O principal objetivo do protótipo é fazer testes de programações, proporções de tela e posição dos sprites, distância da câmera, velocidade dos movimentos e da jogabilidade e analisar a melhor forma de continuação do projeto.

2.4 DESENHOS DOS OBJETOS

Todos os objetos foram desenhados no tablet gráfico e editado no Paint tool SAI, eles foram criados para interagirem diretamente com os personagens do jogo, entre eles destacam-se as árvores e as plataformas.

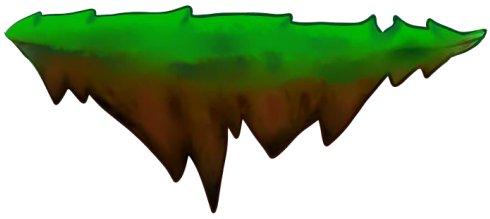
Figura 3: Desenho do sprite: Árvore.

Fonte: Elaborada pelo autor.

Figura 4: Desenho do sprite: Plataforma 1.

Fonte: Elaborada pelo autor.

Figura 5: Desenho do sprite: Plataforma 2.



Fonte: Elaborada pelo autor.

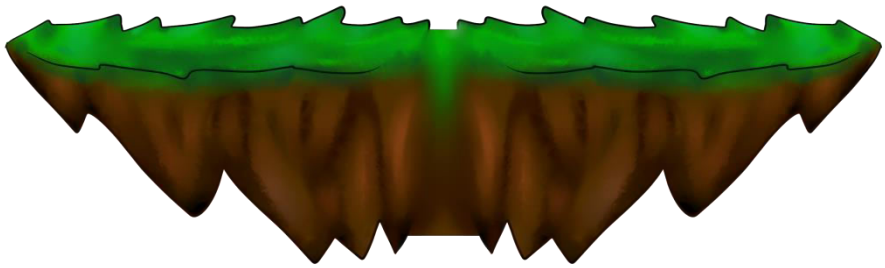
Figura 6: Desenho do sprite: Placa 1.

Fonte: Elaborada pelo autor.

Figura 7: Desenho do sprite: Placa 2.

Fonte: Elaborada pelo autor.

Figura 8: Desenho do sprite: Super plataforma.



Fonte: Elaborada pelo autor.

2.5 PERSONAGENS

Foi decidido criar três personagens neste jogo. Apenas o personagem protagonista é jogável, os outros não são jogáveis; os outros dois personagens são lenhadores que aparecem cortando as árvores da floresta e piratas que aparecem poluindo o oceano.

2.6 DESENHOS DOS PERSONAGENS

Os personagens foram desenhados em estilo mangá no tablet gráfico e editado na ferramenta Paint tool SAI. O personagem protagonista teve o design inspirado em heróis de animação japonesa; os lenhadores possuem o mesmo

design, mas com cores diferentes e foram inspirados em lenhadores reais; os piratas não possuem uma base referencial no design.

Figura 9: Sprite do personagem protagonista.

Fonte: Elaborada pelo autor.

Figura 10: Sprite do personagem lenhador.

Fonte: Elaborada pelo autor.

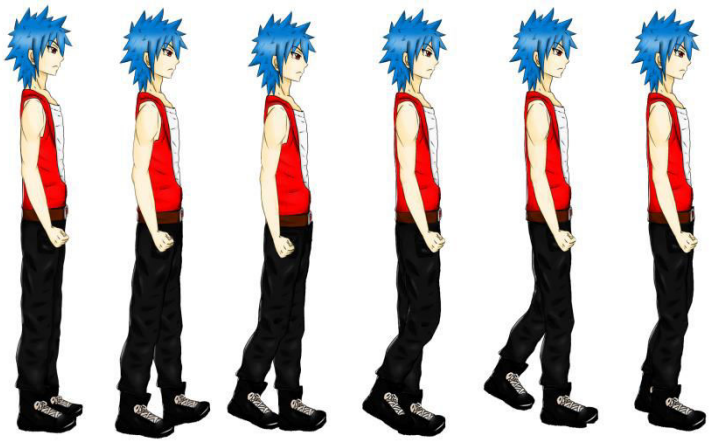
Figura 11: Sprite do personagem pirata.

Fonte: Elaborada pelo autor.

2.7 MOVIMENTAÇÃO DO PERSONAGEM JOGÁVEL

Os comandos do personagem jogável são padrões de jogos de plataforma, onde é possível movimentar para a direita ou esquerda, pular e atacar. Foi configurada a velocidade dos passos e a altura do pulo e seus respectivos comandos pelo teclado.

Figura 12: Sprite do personagem protagonista em movimento.



Fonte: Elaborada pelo autor.

2.8 MOVIMENTAÇÃO DOS PERSONAGENS LENHADORES E PIRATAS

A movimentação dos personagens lenhadores e piratas são automáticas e são ativadas por triggers. O script do lenhador possui uma função para verificar a proximidade do personagem jogável e derrubar a árvore, já o script do pirata verifica a proximidade para atirar no personagem jogável.

2.9 ANIMAÇÕES DE OBJETOS

Alguns objetos possuem animação e movimentos, como as árvores e as plataformas. As árvores possuem animação para transição de inteira de pé para

cortada e caída. As plataformas não possuem animação, mas movimentam de forma vertical no cenário.

Figura 13: Sprites da árvore caindo.

Fonte: Elaborada pelo autor.

2.10 OBJETIVOS DO JOGO

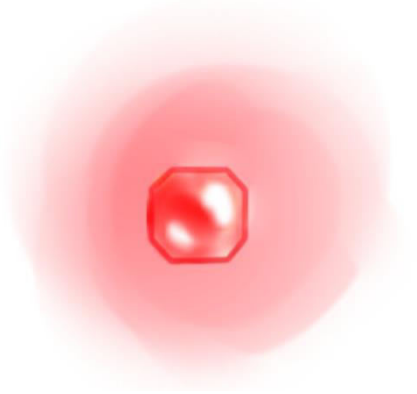
O primeiro objetivo do jogo é explorar toda a fase da floresta e deter os lenhadores para evitar o desmatamento clandestino que está ocorrendo durante a partida, o segundo objetivo é explorar toda a fase da praia e deter os piratas para evitar a poluição no oceano.

2.11 MECÂNICA DE COMBATE E OBJETO DE CURA

O jogador possui um sistema de vida que está presente como uma função do script para o controle das ações. A vida do jogador é representada por números de 1 até 100. O jogador começa com 100 pontos de vida, e caso haja colisão com algum lenhador ou pirata, ele perde de 1 a 5 pontos de vida, ou seja, se o jogador tiver menos de 1 ponto de vida, ele perderá a partida e terá que começar tudo de novo.

Alguns objetos de cura estão presentes em alguns lugares das fases, e é possível recuperar 10 pontos de vida caso o jogador colidir com o objeto de cura.

Figura 14: Objeto de cura.



Fonte: Elaborada pelo autor

3 SISTEMA

O projeto foi desenvolvido utilizando a versão 5.0.0 de 64-bits do Unity, o sistema de vídeo é 32-bits com resolução 1280x720, o formato de tela é 16:9 (Widescreen). A programação é a parte responsável pela jogabilidade do jogo e a linguagem utilizada para escrever os códigos foi C#, uma linguagem de programação orientada a objetos. Os sprites necessitam de scripts que são códigos escritos em C# para programar suas funcionalidades dentro do jogo. Basicamente a programação irá dar vida a quase tudo dentro do jogo.

3.1 CÓDIGOS

Vários scripts foram criados durante o desenvolvimento do projeto; entre eles, PlayerControllerKen.cs, MoveOffsetBG.cs, Lenhador.cs, Tree.cs.

- PlayerControllerKen.cs: Este é o script principal do jogo, responsável pelos comandos de movimentação do personagem, pulo, ataque e também responsável pelo sistema do controle da vida do personagem, ou seja, esse script possibilita controlar o personagem e movimentar o jogo. Esse script possui ligações estáticas com todos os outros scripts, tornando o script mais importante do jogo.
- MoveOffsetBG.cs: Este script é responsável por movimentar o cenário, através das variáveis estáticas, ele se comunica com o PlayerControllerKen.cs para saber se o personagem está em movimento e assim movimentar o cenário.

- Lenhador.cs: Este script é responsável pelos comandos do personagem lenhador, através das variáveis estáticas, ele se comunica com o PlayerControllerKen.cs para saber se o personagem está em movimento, e assim calcular a distância do personagem e acionar o comando para derrubar a árvore.
- Pirata.cs: Este script é responsável pelos comandos do personagem pirata, ele está programado para fazer os piratas andarem normalmente pelo cenário e verificar a distância do jogador para ativar a animação de tiro.
- Tree.cs: Este script possui apenas o sistema do controle da vida da árvore.

3.2 TELAS

Ao executar o jogo, a tela inicial é carregada, o jogador deve clicar na fase para começar o jogo, e as transições de tela são realizadas quando o personagem inicia e termina uma fase. A tela de créditos é executada quando o jogador termina uma fase, e a tela de loading é executada enquanto carrega a fase, apresentando uma breve descrição da fase.

Figura 15: Tela inicial.

Fonte: Elaborada pelo autor.

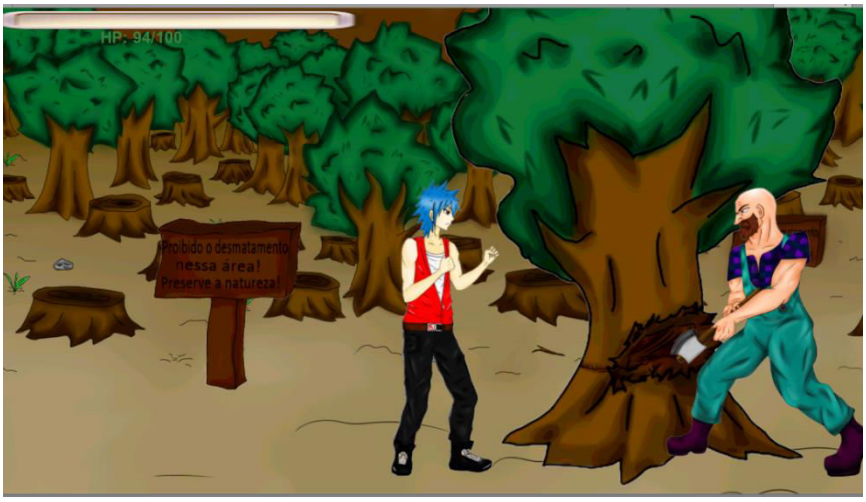
3.3 TESTES

Durante o desenvolvimento deste projeto, vários testes foram realizados para verificar e identificar possíveis falhas que poderiam ocorrer durante o jogo, principalmente quando ocorriam alterações nos códigos ou objetos para corrigir falhas ou adicionar mais conteúdo ao jogo.

3.3.1 Teste: Fase 1 (Forest Area)

Ao executar a primeira fase, os lenhadores estarão acertando as árvores de forma padronizada; o jogador deve atacar o lenhador para fazê-lo parar de tentar derrubar a árvore, mas ele não deve se aproximar muito do lenhador, caso contrário, a árvore irá cair; e para salvar as árvores, o jogador deve atacar à distância, logo, o lenhador estará preso. A fase possui plataformas que estarão se movimentando pela vertical de forma padronizada; o jogador pode pular nas plataformas para explorar toda a fase.

Figura 16: Jogo executando a fase 1.



Fonte: Elaborada pelo autor.

3.3.2 Teste: Fase 2 (Beach Area)

Ao executar a segunda fase, os piratas estarão patrulhando de forma padronizada; o jogador deve atacar todos os piratas que encontrar pelo caminho e não deve se aproximar muito deles, caso contrário o pirata vai atirar e o jogador perderá 5 pontos de vida; o objetivo dessa fase é deter todos os piratas para evitar a poluição de resíduos ilegais causadas por eles. A fase também possui plataformas que se movimentam pela vertical.

Figura 17: Tela de loading da fase 2.

Fonte: Elaborada pelo autor.

Figura 18: Jogo executando a fase 2.

Fonte: Elaborada pelo autor.

4 RESULTADOS E DISCUSSÃO

Neste capítulo serão apresentados alguns resultados obtidos com o desenvolvimento do jogo. O projeto obteve os seguintes resultados:

- O projeto resultou em 52 scripts utilizados.
- O personagem protagonista resultou em 26 sprites no total, tendo 13 sprites para as movimentações à direita e 13 para a esquerda.
- O personagem lenhador resultou em 28 sprites e 7 cores diferentes.
- O personagem pirata resultou em 9 sprites diferentes.
- O objeto árvore resultou em 14 sprites, tendo 12 sprites direcionados para a direita e 12 para a esquerda.
- Foram desenhados 4 tipos diferentes de plataforma, resultando um sprite para cada uma.
- Foram desenhados 2 cenários, resultando em uma textura para cada.
- Foram desenhados alguns objetos aleatórios, entre eles estão o objeto de cura, o tronco, duas placas diferentes e um conjunto de resíduos, resultando em 1 sprite para cada objeto.

Os resultados encontrados mostram o número de scripts e sprites que foram construídos durante o desenvolvimento do jogo. Os scripts são implementados nos sprites para controlar as funcionalidades e o fluxo de animação, portanto, devido ao número de sprites utilizados no projeto, fez-se necessário desenvolver 52 scripts para programar as funcionalidades do jogo. Alguns sprites são utilizados de forma repetida na construção da fase, como no caso das árvores e dos objetos aleatórios, que aparecem vários do mesmo durante a fase, por esse motivo foi necessário criar e programar scripts semelhantes para tal.

CONSIDERAÇÕES FINAIS

O avanço tecnológico traz uma nova realidade para a sociedade atual, é uma ferramenta a mais para a construção de novas percepções e dimensões que buscam nos jogos digitais favorecer o lado educativo. No caso do jogo aqui apresentado, o mesmo fornece uma compreensão dos danos que os seres humanos podem causar ao meio ambiente através das suas ações nocivas.

Todo esse suporte tecnológico revela a necessidade de um envolvimento que resulte em ações positivas pela proposta do uso de jogos digitais como forma de conscientização, uma vez que os mesmos têm conquistado um grande mercado mundial no meio do entretenimento.

A fim de compreender os problemas ecológicos para o desenvolvimento do jogo, fez-se necessário fazer pesquisas relacionadas ao assunto.

Como o objetivo deste projeto é desenvolver um jogo eletrônico sobre conscientização ecológica, o estudo sobre a tecnologia Unity e o desenvolvimento dos desenhos de personagens, cenários e objetos foram essenciais para alcançar os objetivos de construir todo o conteúdo baseado no conceito de conscientização ecológica.

Durante o desenvolvimento, Algumas ferramentas foram utilizadas em conjunto para o propósito final, como para desenhar e colorir os sprites foi utilizado o Paint tool SAI; as transparências no fundo dos sprites foram aplicadas no photoshop. Após a importação para o Unity, os sprites foram animados e programados, atingindo assim as metas para a criação de bons conteúdos que resulte em conscientização dos danos causados pelo desmatamento e pela poluição marinha.

Como o jogo se trata de um software, alguns problemas foram apresentados, logo foram validadas as seguintes hipóteses:

H0: Não seria viável realizar o estudo, pois o desenvolvimento de jogo eletrônico não seria capaz de desenvolver uma conscientização ecológica. **Invalidada**, uma vez que, a pesquisa mostrou que é possível desenvolver um jogo seguindo o conceito de conscientização ecológica, já que, quando o jogo é apresentado de forma conceitual, durante o momento jogado, influencia o desenvolvimento integral e cultual de quem está jogando.

H1: Para desenvolver um jogo eletrônico é necessário saber programar. **Validada**, por mais que o projeto seja simples, é necessário conhecer pelo menos o básico de programação, porque através da programação é construída toda base que dá vida ao jogo.

H2: É possível desenvolver um jogo eletrônico que desperte ou desenvolva uma consciência ecológica.

Validada, é totalmente possível desenvolver um jogo eletrônico que desenvolva no jogador a conscientização dos danos causados à natureza, e que desperte no jogador a consciência de que, enquanto os seres humanos destroem a natureza, eles também estão se autodestruindo, pois todos fazem parte integrante da natureza e que os seus elementos (hidrosfera, atmosfera, litosfera, animais, plantas, entre outros) possui uma relação de interdependência.

Concluiu-se que o resultado final do jogo eletrônico foi satisfatório. As tecnologias Unity, Paint tool SAI e o tablet gráfico são ferramentas muito eficientes para o desenvolvimento de jogos 2D, já que elas possibilitam criar jogos independentes com muita qualidade.

REFERÊNCIAS

CLUA, Esteban Walter Gonzalez; BITTENCOURT, João Ricardo. *Desenvolvimento de Jogos 3D: Concepção, Design e Programação.* 1. ed, São Leopoldo: PUC-Rio, 2005. 49p. Disponível em: <http://www2.ic.uff.br/~esteban/publications.htm>. Acesso em: 07 nov. 2016.

FREIRE, Paulo. *Conscientização: Teoria e Prática da Libertação, uma Introdução ao Pensamento de Paulo Freire*. 1. ed. São Paulo: Cortez & Moraes LTDA, 1979. 53p.

FURLAN, Marcos Paulo. *Photoshop CS6.* 2012. 83p. Disponível em: <http://www.apostilando.com/>. Acesso em: 10 nov. 2016.

LOTAR, Alfredo. *Como Programar com ASP.NET e C#*. 2. ed. São Paulo: Novatec Editora Ltda, 2010. 79p.

MORIMOTO, Carlos E. *Hardware, o Guia Definitivo*. 1. ed. São Paulo: GDH Press e Sul Editores, 2007. 1038p.

NOVAK, J. *Desenvolvimento de Games*. 2. ed. São Paulo: Cergage Learning, 2010.

PEREIRA, Alan Antonio; OLIVEIRA, Tatyanne Freire Duarte de; JUNIOR, Wilson de Oliveira. *Algorhythm, um jogo programado para ensinar a programar.* 2015. Disponível em: <http http://www.aedb.br/seget/artigos2015.php?pag=219 >. Acesso em: 08 nov. 2016.

RESUMOESCOLAR. *Mangá: Origem e características*. Disponível em: <http://www.resumoescolar.com.br/historia/manga-origem-e-caracteristicas/>. Acesso em: 10 nov. 2016.

SILVA, Tiago Aquino da Costa e. *Jogos e brincadeiras na escola.* 1. ed. São Paulo: Kids Move Fitness Programs, 2015. 26p.

UNITY. *Unity - game engine*. 2014. Disponível em: <http://unity3d.com/>. Acesso em: 28 out. 2016.

VIANA, Thiago Campos. *Unity3D: Uma introdução*. 1. ed. São Carlos-SP, 2009. 56p.

VINCENT. *Informaçoes Basicas e Caracteristicas sobre o mangá.* 2016. Disponível em: <http://asasdaanimacao.blogspot.com.br/2016/05/informacoes-basicas-e caracteristicas.html>. Acesso em: 10 nov. 2016.

WATKINS, A. *Criando Jogos Com Unity e Maya - Como Desenvolver Jogos 3D Divertidos e de Sucesso*. 1. ed. São Paulo: Ed Campus, 2012.

ANEXOS

ANEXO 1: Script completo para controle das ações do jogador

```
using UnityEngine;
using System.Collections;

public class PlayerControllerKen : MonoBehaviour {


	public Animator Anime;
	public Rigidbody2D playerRigidbody;
	public int forceJump;

	public static bool Cp;
	public static bool CpUp;
	public static bool punchRight;
	public static bool right;
	public static bool normalRight;
	public static bool punchLeft;
	public static bool left;
	public static bool normalLeft;
	public Transform GroundCheck;
	public static bool grounded;
	public LayerMask whatIsGround;

	public static bool jump;
	public static bool jumpLeft;

	public float punchRightTemp;
	public float rightTemp;
	public float punchLeftTemp;
	public float leftTemp;
	public float timeTemp;

	private float y;
	private float x;
	private float z;

	public Transform colisor;

	public AudioSource audio;
	public AudioClip soundJump;
	public AudioClip soundPunch;

	public Transform plataform;


	public GameObject ataqueEspecial;
	public GameObject ataqueEspecialLeft;
	public float duracaoAtaque;
	private float contagemAtaque;
```

```
	public static bool colidir;
	public float velocidade;

	public static GUIText guiText;
	public static int maxVida = 100;
	public static int vidaAtual;
	public GUIText vida;

	// Use this for initialization
	void Start () {
		guiText = vida;
		vidaAtual = maxVida; /* Ao iniciar a fase, a vida atual do personagem será igual ao máximo que ele pode ter de vida, ou seja, estará com o HP completo.*/
		guiText.color = new Vector4(0.25f, 0.5f, 0.25f, 1f); /* Dentro do objeto vida, estamos entrando no componente GUIText e na variável color, mudando esse valor para verde.*/

		guiText.text = "HP: " + vidaAtual + "/" + maxVida; /*  estou dizendo que o campo Text do objeto vida no componente GUIText vai receber “HP 100/100”, caso vidaAtual seja 100 e maxVida seja 100.*/
		normalRight = true; // Normalmente parado é verdadeiro

	}

	// Update is called once per frame
	void Update () {
		// SE O BOTÃO FOR APERTADO E TIVER PISANDO NO CHÃO, ENTÃO FAÇA ISSO
		if(Input.GetButtonDown ("Jump") && grounded == true) {
			playerRigidbody.AddForce (new Vector2 (0, forceJump)); // ADICIONA FORÇA AO PULO
			audio.PlayOneShot(soundJump); // ADICIONA SOM AO PULO
			audio.volume = 1; // AJUSTA O VOLUME
		}

		// SE O BOTÃO FOR APERTADO E TIVER PISANDO NO CHÃO A ESQUERDA, ENTÃO FAÇA ISSO
		if(Input.GetButtonDown ("jumpLeft") && grounded == true) {
			playerRigidbody.AddForce (new Vector2 (0, forceJump)); // ADICIONA FORÇA AO PULO
			audio.PlayOneShot(soundJump); // ADICIONA SOM AO PULO
			audio.volume = 1; // AJUSTA O VOLUME
		}

		// SE O BOTÃO FOR APERTADO E TIVER PISANDO NO CHÃO PELA DIREITA E SOCO FOR FALSO E TIVER NORMALMENTE PARADO, ENTÃO FAÇA ISSO
		if(Input.GetButtonDown("punchRight") && grounded == true && punchRight == false && normalRight == true){
			// O COLISOR RECEBERÁ UM NOVO VETOR DE 3
```

```
POSIÇÕES E VAI MANTER A MESMA POSIÇÃO QUE JÁ ESTÁ
                        colisor.position = new Vector3(colisor.position.x,
colisor.position.y, colisor.position.z);
                        punchRight = true; // ATAQUE SERÁ VERDADEIRO
                        normalLeft = false; //  ATAQUE SERÁ FALSO
                        timeTemp = 0; // CONTAGEM DO ATAQUE INICIA EM 0
                        audio.PlayOneShot(soundPunch); // SOM DO ATAQUE
                        audio.volume = 0.23f; // VOLUME DO ATAQUE
                        Power.poder = false; // VARIÁVEL AUXILIAR PARA O ATAQUE
ESPECIAL É FALSA
                        ataqueEspecial.SetActive(true); // ATAQUE ESPECIAL SERÁ
ATIVADO
                        }
                // SE O BOTÃO FOR APERTADO E TIVER PISANDO NO CHÃO PELA
ESQUERDA E SOCO FOR FALSO E TIVER NORMALMENTE PARADO, ENTÃO
FAÇA ISSO
                if(Input.GetButtonDown("punchLeft") && grounded == true &&
punchLeft == false && normalLeft == true){
                        // O COLISOR RECEBERA UM NOVO VETOR DE 3
POSIÇÕES E VAI MANTER A MESMA POSIÇÃO QUE JÁ ESTÁ
                        colisor.position = new Vector3(colisor.position.x,
colisor.position.y, colisor.position.z);
                        punchLeft = true; //  ATAQUE SERÁ VERDADEIRO
                        punchRight = false; //  ATAQUE SERÁ FALSO
                        timeTemp = 0; //  CONTAGEM DO ATAQUE INICIA EM 0
                        audio.PlayOneShot(soundPunch); // SOM DO ATAQUE
                        audio.volume = 0.23f; // VOLUME DO ATAQUE
                        PowerLeft.poder = false; // VARIÁVEL AUXILIAR PARA O
ATAQUE ESPECIAL É FALSA
                        ataqueEspecialLeft.SetActive(true); // ATAQUE ESPECIAL
SERÁ ATIVADO
                }

                // ENQUANTO O BOTÃO ESTIVER APERTADO E TIVER PISANDO
NO CHÃO E NÃO ESTIVER MOVIMENTANDO PARA A DIREITA
                if(Input.GetAxis("Horizontal") == 1 && grounded == true && right ==
false){
                        // O COLISOR RECEBERA UM NOVO VETOR DE 3
POSIÇÕES E VAI MANTER A MESMA POSIÇÃO QUE JÁ ESTÁ
                        colisor.position = new Vector3(colisor.position.x,
colisor.position.y, colisor.position.z);
                        right = true; // MOVIMENTAÇÃO PARA A DIREITA SERÁ
VERDADEIRA
                        normalRight = false; // NORMALMENTE PARADA PARA A
DIREITA SERÁ FALSO
                        timeTemp = 0; // CONTAGEM DO MOVIMENTO INICIA EM 0
                        }

                // SE O BOTÃO FOR DESAPERTADO E TIVER PISANDO NO CHÃO
E NÃO ESTIVER NORMALMENTE PARADO PARA A DIREITA
```

```
                if(Input.GetButtonUp("Horizontal") && grounded == true && normalRight
== false){
                        // O COLISOR RECEBERA UM NOVO VETOR DE 3
POSIÇÕES E VAI MANTER A MESMA POSIÇÃO QUE JÁ ESTÁ
                        colisor.position = new Vector3(colisor.position.x,
colisor.position.y, colisor.position.z);
                        normalRight = true; // NORMALMENTE PARADO PARA A
DIREITA SERÁ VERDADEIRO
                        right = false; // MOVIMENTAÇÃO PARA A DIREITA SERÁ
FALSA
                        }

                // ENQUANTO O BOTÃO ESTIVER APERTADO E TIVER PISANDO
NO CHÃO E NÃO ESTIVER MOVIMENTANDO PARA A ESQUERDA
                if(Input.GetAxis("Horizontal") == -1 && grounded == true && left ==
false){
                // O COLISOR RECEBERÁ UM NOVO VETOR DE 3 POSIÇÕES E VAI
MANTER A MESMA POSIÇÃO QUE JÁ ESTÁ
                        colisor.position = new Vector3(colisor.position.x,
colisor.position.y, colisor.position.z);
                        left = true; // MOVIMENTAÇÃO PARA A ESQUERDA SERÁ
FALSA
                        normalLeft = false; // NORMALMENTE PARADO PARA A
ESQUERDA SERÁ VERDADEIRO
                        timeTemp = 0; //  CONTAGEM DO MOVIMENTO INICIA EM 0
                }

                // SE O BOTÃO FOR DESAPERTADO E TIVER PISANDO NO CHÃO
E NÃO ESTIVER NORMALMENTE PARADO PARA A ESQUERDA
                if(Input.GetButtonUp("Horizontal") && grounded == true && normalLeft
== false){
                        // O COLISOR RECEBERÁ UM NOVO VETOR DE 3
POSIÇÕES E VAI MANTER A MESMA POSIÇÃO QUE JÁ ESTÁ
                        colisor.position = new Vector3(colisor.position.x,
colisor.position.y, colisor.position.z);
                        normalLeft = true; // NORMALMENTE PARADO PARA A
ESQUERDA SERÁ VERDADEIRO
                        left = false; // MOVIMENTAÇÃO PARA A ESQUERDA SERÁ
FALSA
                        normalRight = false; // NORMALMENTE PARADO PARA A
DIREITA SERÁ VERDADEIRO
                }
                grounded = Physics2D.OverlapCircle (GroundCheck.position, 0.2f,
whatIsGround); // VERIFICA O QUE É CHÃO

// SE O JOGADOR E AS PLATAFORMAS COLIDIREM, O JOGADOR NÃO IRÁ
PISAR NA PLATAFORMA ENQUANTO A VELOCIDADE DO PULO FOR MAIOR
QUE 0.
Physics2D.IgnoreLayerCollision(LayerMask.NameToLayer("PlayerKen"),LayerMask.
NameToLayer("Ground2"), rb.velocity.y > 0);
```

```
            // SE ATAQUE A DIREITA FOR VERDADEIRO, ENTÃO FAÇA
            if (punchRight == true) {

                timeTemp += Time.deltaTime; // O TEMPO QUE PASSOU
                // SE O TEMPO DO ATAQUE A DIREITA FOR MAIOR OU
IGUAL O TEMPO QUE PASSOU
                if (timeTemp >= punchRightTemp) {
                    // O COLISOR RECEBERÁ UM NOVO VETOR DE 3
POSIÇÕES E VAI MANTER A MESMA POSIÇÃO QUE JÁ ESTÁ
                    colisor.position = new Vector3(colisor.position.x,
colisor.position.y, colisor.position.z);
                    punchRight = false; // ATAQUE A DIREITA SERÁ FALSO
            }
            }
            // SE SOCO A ESQUERDA FOR VERDADEIRO, ENTÃO FAÇA
            if (punchLeft == true) {

                timeTemp += Time.deltaTime; // O TEMPO QUE PASSOU
                // SE O TEMPO DO ATAQUE A ESQUERDA FOR MAIOR OU
IGUAL O TEMPO QUE PASSOU
                if (timeTemp >= punchLeftTemp) {
                    // O COLISOR RECEBERÁ UM NOVO VETOR DE 3
POSIÇÕES E VAI MANTER A MESMA POSIÇÃO QUE JÁ ESTÁ
                    colisor.position = new Vector3(colisor.position.x,
colisor.position.y, colisor.position.z);
                    punchLeft = false; // ATAQUE A ESQUERDA SERÁ
FALSO
                }
            }
            // SE A MOVIMENTAÇÃO A DIREITA FOR VERDADEIRA, ENTÃO
FAÇA
                if (Power.poder == true) {
                ataqueEspecial.SetActive(false); // ATAQUE ESPECIAL É
FALSO
            }
            // SE A VARIÁVEL AUXILIAR DE ATAQUE ESPECIAL A ESQUERDA
FOR VERDADEIRO, ENTÃO FAÇA
            if (PowerLeft.poder == true) {
            ataqueEspecialLeft.SetActive(false); // ATAQUE ESPECIAL É FALSO
            }
            if (Cp == true && PlataformaMoveRight.velocidade == 0) { // SE A
VARIÁVEL TEMPORÁRIA FOR VERDADEIRA E VELOCIDADE DA PLATAFORMA
FOR ZERO
                x = plataform.position.x; // X TERÁ O VALOR DA POSIÇÃO DA
PLATAFORMA
            transform.position = new Vector3 (x, transform.position.y,
colisor.position.z); // A POSIÇÃO X TERÁ UMA NOVO VALOR
            }
             // SE TODAS AS ARVORES FOREM SALVA, ENTÃO
```

```
                if(Tree.fallenTree == true && Tree1.fallenTree == true &&
Tree2.fallenTree == true && Tree3.fallenTree == true && Tree4.fallenTree == true &&
Tree5.fallenTree == true && Tree6.fallenTree == true && Tree8.fallenTree == true){
                Application.LoadLevel("GameOver"); // O JOGO TERMINA E VAI
APARECER A TELA DE CRÉDITOS
                }
                // SE 19 OU MAIS PIRATAS FOREM DERROTADOS, ENTÃO
                if(Pirata.f >= 19 && Chegou.chegou == true){
                        Application.LoadLevel("GameOver"); // O JOGO TERMINA E VAI
APARECER A TELA DE CRÉDITOS
                }
                Anime.SetBool("jump", !grounded); // A ANIMAÇÃO DE PULO SERÁ
ATIVADA QUANDO NÃO ESTIVER PISANDO NO CHÃO
                Anime.SetBool("jumpLeft", !grounded); // A ANIMAÇÃO DE PULO A
ESQUERDA SERÁ ATIVADA QUANDO NÃO ESTIVER PISANDO NO CHÃO
                Anime.SetBool ("punchRight", punchRight); // A ANIMAÇÃO DE SOCO
A DIREITA SERÁ ATIVADA QUANDO APERTAR BOTÃO
                Anime.SetBool ("right", right); // A ANIMAÇÃO DE MOVIMENTAÇÃO A
DIREITA SERÁ ATIVADA QUANDO APERTAR O BOTÃO
                Anime.SetBool ("normalRight", normalRight); // A ANIMAÇÃO DE
NORMALMENTE PARADO A DIREITA SERÁ ATIVADO QUANDO DESAPERTAR
O BOTÃO
                Anime.SetBool ("punchLeft", punchLeft); // A ANIMAÇÃO DE SOCO A
ESQUERDA SERÁ ATIVADA QUANDO APERTAR BOTÃO
                Anime.SetBool ("left", left); // A ANIMAÇÃO DE MOVIMENTAÇÃO A
ESQUERDA SERÁ ATIVADA QUANDO APERTAR O BOTÃO
                Anime.SetBool ("normalLeft", normalLeft); // A ANIMAÇÃO DE
NORMALMENTE PARADO A ESQUERDA SERÁ ATIVADO QUANDO
DESAPERTAR O BOTÃO
        }
        // FUNÇÃO PARA COLISÃO
        void OnCollisionEnter2D(Collision2D other){
                left = false; // movimentação para esquerda será falsa se colidir com
algo
                right = false; // movimentação para direita será falsa se colidir com algo
                if (left == false) { // se movimentação a esquerda for falso
                normalLeft = true; // normalmente parado a esquerda será verdadeiro
                }
                if (right == false) { // se movimentação a direita for falso
                normalRight = true; // normalmente parado a direita será verdadeiro
                }
        }
        // função para diminuir os pontos de vida
        public static void PerdeVida(int dano) {
                vidaAtual -= dano; // Nessa linha dizemos que a vida atual do
personagem vai ser reduzida pelo dano que ele levou.
                if (vidaAtual <= 0) { // Se a vida atual do personagem for zero ou menor
que isso        animator.SetTrigger("Falling"); // Vai executar a animação de cair
                 }else{ // Se não morrer
                 animator.SetTrigger("pain"); // Vai executar a animação de dor
```

```
                }
                 if (vidaAtual < 0){ // Se a vida for menor que 0, então
                Application.LoadLevel(Application.loadedLevel); // reiniciará a fase
                }
                if ((vidaAtual * 100 / maxVida) < 30) { // e a vida do personagem estiver
abaixo de 30%
                guiText.color = Color.red; // a cor dos pontos de vida ficará vermelha
                }
                guiText.text = "HP: " + vidaAtual + "/" + maxVida; /* Altera o texto do
componente GUIText do Objeto Vida para “HP: 90/100” se ele levou um dano de 10
e estava com a vida cheia.*/
        }
        // função para recuperar os pontos de vida
        public static void RecuperaVida(int recupera) {
                vidaAtual += recupera; /* A vida atual do personagem irá receber um
valor a mais de acordo com o valor passado no parâmetro recupera na declaração
do método RecuperaVida(int recupera).*/
                if (vidaAtual > maxVida) { // Como a vida atual não pode ser maior que
a vida máxima
                vidaAtual = maxVida; // alteramos a vida atual para o mesmo valor de
maxVida deixando: 100/100.
                }
                if ((vidaAtual * 100 / maxVida) >= 30) { // se a vida do personagem
estiver acima de 30%
                guiText.color = new Vector4(0.25f, 0.5f, 0.25f, 1f); // A cor dos pontos
de vida ficara verde
                }
                guiText.text = "HP: " + vidaAtual + "/" + maxVida; // Atualiza o texto do
objeto Vida.
        }
}
```

PlayerControllerKen.cs

- Time.deltaTime é quanto tempo passou entre o movimento e outro.
- timeTemp = 0; Esta variável recebe 0 para garantir que vai começar uma contagem nova.
- timeTemp += Time.deltaTime; o tempo do movimento é somado com o tempo passou entre o movimento e outro; isso é utilizado para os frames se igualarem independente da plataforma que o jogo esteja sendo executado.
- playerRigidbody.AddForce (new Vector2 (0, forceJump)); Está função permite o personagem pular; O Rigidbody é um componente para física do jogo, no caso dessa função, o componente está adicionando força ao personagem utilizando o método AddForce para executar o pulo e passar a nova posição para o vetor.

ANEXO 2: Script completo para movimentação do cenário

```
using UnityEngine;
using System.Collections;

public class MoveOffsetBG : MonoBehaviour {

        private Material currentMaterial;
        public static float speed;
        private float offSet;
        public static float x;
        public GameObject PlayerKen;
        public float timeTemp;

        // Use this for initialization
        void Start () {
        // O currentMaterial vai armazenar a textura utilizada no objeto que esta o
script
                currentMaterial = GetComponent<Renderer> ().material;
        }
        // Update is called once per frame
        void Update () {
        // Se a movimentação a direita do personagem for verdadeira
        if (PlayerControllerKen.right == true) {
        // mantém o valor da variável offset e somar com a velocidade de
movimento e multiplica com o Time.deltaTime
        // o Time.deltaTime é quanto tempo passou entre o movimento e outro
        offSet += speed * Time.deltaTime;
        // aplica o valor da multiplicação no currentMaterial
        currentMaterial.SetTextureOffset ("_MainTex", new Vector2 (offSet, 0));
                speed = 0.10f; // Definindo a velocidade do movimento da textura
                }
        // Se a movimentação a esquerda do personagem for verdadeira
        if (PlayerControllerKen.left == true) {
        offSet += speed * Time.deltaTime;
        currentMaterial.SetTextureOffset ("_MainTex", new Vector2 (offSet, 0));
                speed = -0.10f;
                }
        // Se a colisão do personagem com a plataforma for verdadeira
        if (PlayerControllerKen.Cp == true) {
        offSet += speed * Time.deltaTime;
        currentMaterial.SetTextureOffset ("_MainTex", new Vector2 (offSet, 0));
        x = PlayerKen.transform.position.x; // O valor horizontal da textura será igual
o do personagem
        // O vetor horizontal recebe um novo valor
        transform.position = new Vector3 (x, transform.position.y,
transform.position.z);
        }
    }
}
```

MoveOffsetBG.cs