EDITOR DE CENÁRIOS SIMULADOS EM 3D

Bruno Fischer Ferreira Santos, Dalton Solano dos Reis – Orientador

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

bffsantos@furb.br, dalton@furb.br

Resumo: *Este artigo descreve o desenvolvimento de um editor de cenários simulados junto ao estudo de técnicas para construção das suas funcionalidades. O objetivo é implementar um editor de superfícies 3D para criação de terrenos virtuais. Desta forma se busca trazer o ambiente do mundo real para do virtual apresentando simulação do ciclo da água e construção de paisagens. Para representar algumas das características presentes no mundo utiliza-se técnicas de construção de malhas triangulares constituindo regiões hídricas, uso de Trails na criação de estradas, fluxos de água através do sistema de partículas e simulação de inundação e evaporação. A aplicação destes procedimentos validou algumas das técnicas na sua maioria em cenários controlados como construção de malhas e inundação de regiões, sendo necessário um estudo mais aprofundado dos métodos para contemplar cenários não controlados. Por fim, algumas técnicas mostraram potencial como estudo da malha tanto hídrica quanto malha das estradas e outras, como as simulações, necessitam aprimoramentos.*

Palavras-chave*: Terrenos virtuais. Editor de cenários. Malhas poligonais. TrailRenderer. Simulações.*

1 INTRODUÇÃO

Em todos os períodos da história, o relevo sempre é um obstáculo nas escolhas e adaptações humanas. De acordo com Pena (2018, p.1), "relevo e sociedade formam diferentes faces de uma mesma composição estrutural, a qual responde pela interação das atividades humanas com a cadeia de elementos naturais". Considerado esta relação, relevo e sociedade, é observado o quanto o meio natural condiciona as atividades humanas (PENA, 2018). Desta maneira, para qualquer planejamento estrutural se faz necessário um estudo do relevo local.

O relevo é a forma da superfície terrestre e pode-se destacar quatro tipos: planície, montanha, depressão e planalto (FRANCISCO, 2018). Francisco (2018) identifica planícies sendo terrenos relativamente planos próximos ao nível do mar, montanhas como grandes elevações da superfície terrestre, depressões caracterizadas por altitudes inferiores ao relevo em sua volta e planaltos como relevos marcados pela sua variação de altitude. Para entendimento da formação destas formas de relevo, simuladores são desenvolvidos a fim de reproduzir o comportamento do sistema entre relevos e edificações do mundo real.

O simulador é um dispositivo que reproduz suas próprias condições de uma determinada atividade, em outras palavras, um sistema técnico que imita circunstâncias reais (QUECONCEITO, 2017). Na área de simuladores de terrenos e superfícies há o exemplo da ferramenta Google Earth, que iniciou disponibilizando mapas viários sobre fotos aéreas reais dispostos no formato do globo terrestre até trazer, em versões mais recentes, ondulações no relevo e representações tridimensionais de cidades e das maiores engenharias da humanidade. Esta visualização só se fez concreta com a construção do ambiente gráfico a partir de um editor de superfície. Tais ambientes englobam a modelagem de relevo, arborização e disposição de edificações. A representação de terrenos e a construção de mapas podem ser utilizadas para planejamento e simulação de trajetória, identificação de pontos de contato, análise de estabilidade, entre outras (SANGREMAN; FREITAS; COSTA, 2013).

Neste contexto, este trabalho desenvolveu um editor de superfícies 3D para criação de terrenos virtuais, facilitando a visualização do relevo e planejamento na construção de um cenário simulado. Os objetivos específicos são: importar o relevo de um terreno real para o terreno virtual, simular terrenos virtuais com características próximas a de terrenos reais, permitir adicionar edificações e pavimentações.

2 FUNDAMENTAÇÃO TEÓRICA

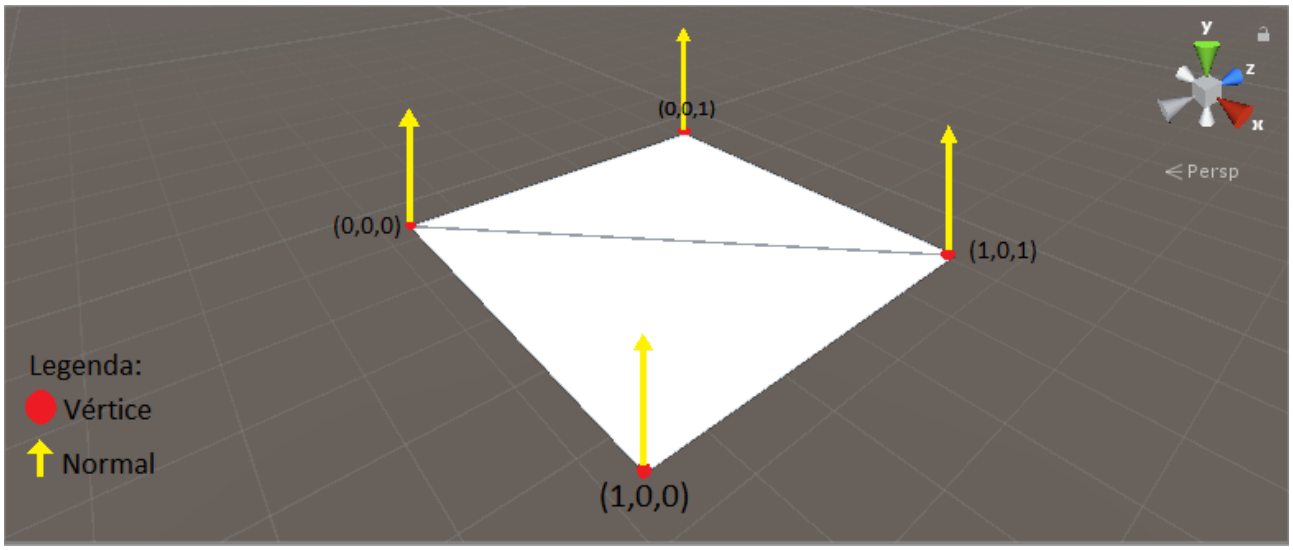
Este capítulo é composto pelos principais conceitos e técnicas estudados e aplicados no desenvolvimento deste trabalho. É apresentado sobre malhas triangulares e seu modo de construção e sobre `TrailRenderer` e seu potencial na criação de malhas, ambos usando o ambiente de desenvolvimento Unity. Ao final é descrito trabalhos correlatos a este desenvolvido como: Terrain.party um gerador de arquivos de mapa de altura; modo de edição de cenário do jogo Age of Empires II e o Craftscape, uma aplicação Web que simula a erosão hidráulica.

2.1 MESH

De acordo com o Unity (2019a), `Mesh` é um conjunto de triângulos dispostos de forma organizada no ambiente tridimensional resultando em um objeto sólido, sendo a principal forma de representação gráfica da ferramenta. Para se observá-lo no plano é necessário que o *GameObject* contenha dois componentes, `MeshFilter` e `MeshRenderer`. O `MeshFilter` é a interface que contém a referência do `Mesh` e suas propriedades, como vértices e triângulos. O `MeshRenderer` é o componente que processa os dados geométricos do `MeshFilter` e torna o modelo do objeto visível (UNITY, 2019c).

Existem duas formas de se obter uma malha no Unity, importando um modelo pronto construído em uma ferramenta externa ou criando-a via script. O mínimo necessário para a implementação de uma malha via script é determinar vértices e triângulos. Unity (2019a, p.1, tradução nossa) alega que "os vértices são atribuídos em um único vetor e os triângulos são especificados a partir de três números inteiros condizentes ao índice do vetor dos vértices". Os dados contidos na matriz de vértices são suas coordenadas x, y e z do vértice no espaço do mundo e o vetor de triângulos contêm números inteiros, que de acordo com o Unity (2019e), as três primeiras posições correspondem ao primeiro triângulo, as próximas três posições ao segundo e assim sucessivamente. A Figura 1 apresenta as coordenadas de cada vértice.

Figura 1 – Representação das propriedades em uma malha

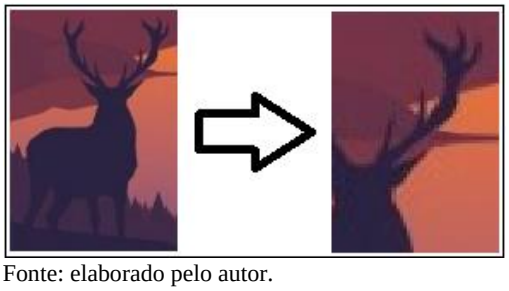


Fonte: elaborado pelo autor.

Para a correta ambientação no espaço 3D, o `Mesh` contém duas propriedades, os `Normals` e aplicação de textura. `Normals` são vetores perpendiculares a face do triângulo localizados em cada vértice em que está relacionado (UNITY, 2019a). Desta maneira ao se relacionar com a direção da luz, que também é um vetor, calcula-se o ângulo de entrada da luz no vetor `Normal` resultando em um sombreamento no objeto (UNITY, 2019a). A Figura 1 mostra a direção do vetor Normal em uma malha quadrada simples.

Além disto, é possível ganhar detalhes com texturas, podendo fazê-las especificamente para um modelo pronto ou configurando através de script. É estabelecida uma área triangular na imagem de textura, ajustando-a nas dimensões do triângulo da malha (UNITY, 2019a). Este procedimento utiliza coordenadas, chamadas de U e V, bidimensionais e dimensionadas em um intervalo entre 0 e 1, onde 0 significa a parte inferior esquerda e 1 a parte superior direita da imagem (UNITY, 2019a). Na Figura 2 é exibido o recorte da textura com valor de U valendo 0.5 e V valendo 1. A causa de serem chamadas de U e V é para não serem confundidas com o X e Y do mundo 3D (UNITY, 2019a).

Figura 2 – Aplicação da coordenada de textura



Fonte: elaborado pelo autor.

2.2 TRAILRENDERER

O `TrailRenderer` é um componente do Unity que tem como função criar um rastro atrás do objeto assim que ele se move (UNITY, 2019d). Este rastro é um aglomerado de polígonos que formam uma malha, tendo as configurações baseadas na largura, comprimento, tempo de vida e textura destes polígonos (UNITY, 2019d). O rastro é bidimensional e tem como características ser sempre rotacionado com a face virada para a câmera (UNITY, 2019d).

Este componente tem uma propriedade chamada `Min Vertex Separation`. Esta propriedade define a distância do último vértice mais próximo do objeto em relação a quando o próximo polígono será adicionado na malha do rastro (UNITY, 2019d). Portanto, valores mais próximos de zero criam rastro mais detalhados e suaves e valores mais altos deixam o rastro mais intervalado (UNITY, 2019d). O valor deste recurso é importante para otimizar a iluminação sobre o rastro, pois o valor de sombreamento é calculado através dos vértices da malha e vértices muito próximos podem resultar em uma iluminação incorreta ou em uma textura deformada.

Para contornar o problema com a distância dos vértices o `TrailRenderer` usa `Materials` com `Particle Shaders`. A renderização de um modelo é definida por um `Material` que contém é a referência de uma textura no Unity (UNITY, 2019b). `Particle Shader` é um script que pode ser aplicado ao `Material`. Este script usa da iluminação sobre a malha do objeto e cálculos matemáticos para definir a cor de cada pixel da textura (UNITY, 2019b).

Figura 3 – Visualização do TrailRenderer

Fonte: Unity (2019).

2.3 RAYCASTING

De acordo com Glover (2017, p.1, tradução nossa) “Raycasting é o processo de traçar um raio invisível a partir de um ponto, em uma direção específica para detectar se algum Collider está no caminho do raio”. Para o mínimo funcionamento do `Raycast` é necessário definir um ponto de origem e a direção do raio em partindo da origem. Além destes parâmetros há outros como a distância máxima que o segmento vai ser disparado, as camadas que o raio traçado pode ignorar o evento de colisão e os gatilhos que não devem ser acionados em caso de colisão.

No Quadro 1 é apresentado um trecho de código que contém as estruturas necessárias para coletar informações de um ponto usando a `API Physics.Raycast`. A finalidade deste exemplo é criar um raio partindo do cursor do mouse em direção ao espaço do mundo 3D, porém o cursor não contém uma coordenada do mundo. Utilizando o método `ScreenToRay()` passando a posição do mouse como parâmetro converte-se a posição do mouse em pixels para coordenada do espaço do mundo. Estes dados, origem e direção, são armazenados na estrutura `Ray`. É também declarado uma variável do tipo `RaycastHit` que é por onde as informações do ponto de colisão são armazenadas. O teste de colisão é feito na linha 6 a partir do método `Physics.Raycast()`. O primeiro parâmetro é onde contém a origem e a direção do raio, o segundo é as informações obtidas em caso de colisão e o último é a distância máxima que o raio pode ser traçado. Na linha 7, através do `Debug.DrawLine()`, se desenha uma linha da origem até o ponto de colisão.

Quadro 1 – Exemplo de implementação do `Raycast`

```
1  void Update()
2      {
3          Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
4          RaycastHit hit;
5
6          if (Physics.Raycast(ray, out hit, 100))
7              Debug.DrawLine(ray.origin, hit.point);
8      }
```

Fonte: elaborado pelo autor.

2.4 TRABALHOS CORRELATOS

Nesta seção é apresentado três trabalhos correlatos que apresentam conceitos e ou funcionalidades semelhantes descritas neste trabalho. O primeiro correlato é a ferramenta terrain.party (2015) e é inicialmente apresentado no Quadro 2. Em sequência no Quadro 3 é apresentado o editor de cenários do jogo Age of Empires II (1999). Por último no Quadro 4 a aplicação Craftscape (2011), que simula erosões hidráulicas.

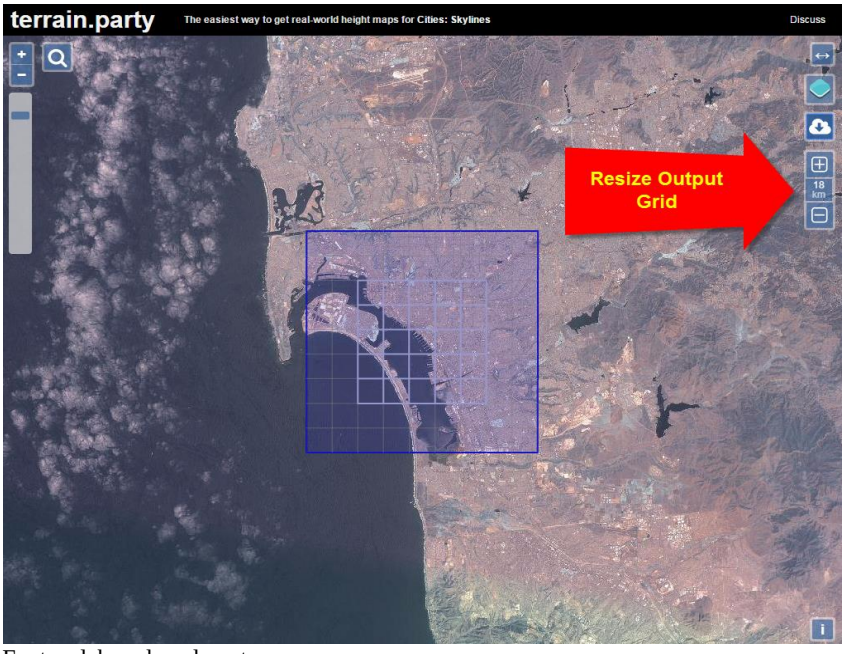
Quadro 2 – Terrain.party

Referência	(2015)
Objetivos	Utilizar terrenos do mundo real dentro do jogo Cities Skylines.
Principais funcionalidades	A ferramenta possibilita selecionar uma região entre 8km² a 60km². Está região é transformada em cinco mapas de altura que diferenciam na sua qualidade.
Ferramentas de desenvolvimento	Não encontrado.
Resultados e conclusões	O formato das porções de terra selecionadas se mantém quando passada a região para o jogo. Porém, a precisão não é constante para qualquer região. Quando selecionado locais com água, há um risco de que no momento de importação a ferramenta não ofereça uma precisão ideal das alturas. As regiões litorâneas não possuem a profundidade real, apenas são identificadas por uma região íngreme.

Fonte: elaborado pelo autor.

Os ajustes na altura do relevo são feitos na escala 1/64 avos, sendo por exemplo, 64 metros reais o equivalente a 1 metro no jogo. Somente a visão viária cobre o mundo inteiro, para as outras duas opções, topográfica e sombreada, apenas há América do Norte, América Central, norte da América do Sul e regiões muito pequenas da Ásia e Europa. Os mapas são disponibilizados sob a licença Open Data Commons Open Database License pela Fundação OpenStreetMap, onde ela dá o direito de utilizar seus mapas desde que atribua a sua autoria. A Figura 4 mostra o processo de seleção da região.

Figura 4 – Seleção da região em terrain.party



Fonte: elaborado pelo autor.

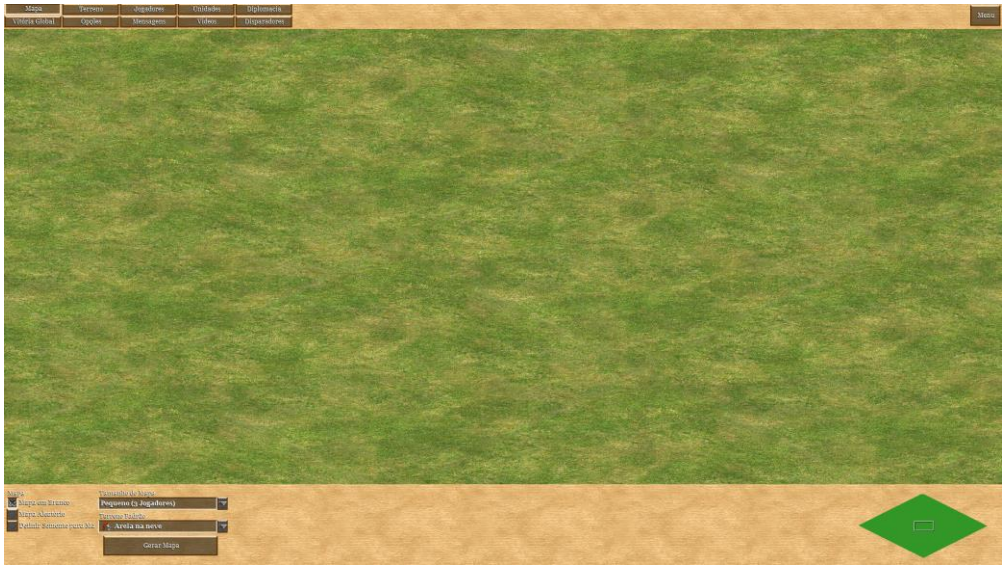
Quadro 3 – Editor de cenário do jogo Age of Empires II

Referência	Ensemble Studios (1999)
Objetivos	Criar cenários jogáveis para o jogo Age of Empires II.
Principais funcionalidades	Sua ferramenta de edição de cenário é composta de funcionalidades de deformação de relevo, demarcação de regiões marinhas, disposição de recursos, disposição de unidades, disposição de construções.
Ferramentas de desenvolvimento	Não encontrado.
Resultados e conclusões	O que geralmente ocorre é de que a ferramenta disponibilizada no jogo para criação de cenários é a mesma usada no desenvolvimento do jogo.

Fonte: elaborado pelo autor.

A Figura 5 apresenta a interface do editor do jogo Age of Empires II. No canto superior esquerdo contém um conjunto de botões que selecionam um modo de edição como modificação de terreno, posicionamento de unidades, aplicação de textura, entre outros. No canto inferior esquerdo encontra-se um painel de configuração do modo selecionado. Do lado direito na parte inferior o mapa do cenário e em casos de disposição de edifícios e unidades abre-se uma lista contendo todos os objetos do jogo no canto esquerdo da tela.

Figura 5 – Interface do editor de cenário do jogo Age of Empires II



Fonte: elaborado pelo autor.

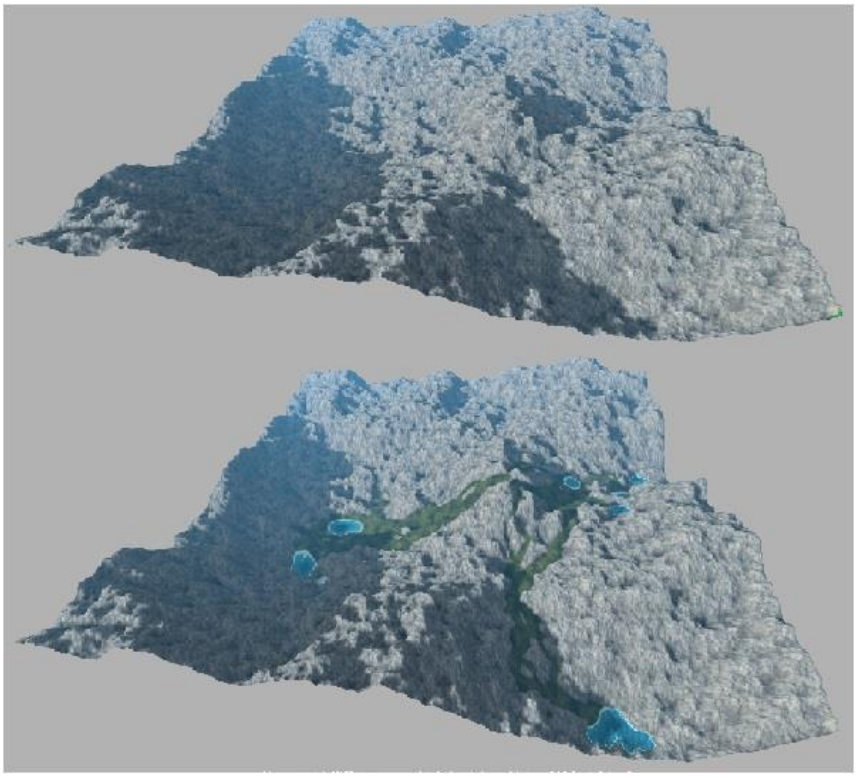
Quadro 4 – Craftscape

Referência	Boesch (2011)
Objetivos	Simular erosão hidráulica.
Principais funcionalidades	É possível habilitar 3 estados no ambiente de erosão, chuva e evaporação. Pode ser criado um fluxo de água e remover focos de água acumulada. Há opção de trocar o *seed* para gerar um novo terreno.
Ferramentas de desenvolvimento	Aplicação foi desenvolvida para WebGL em HTML5.
Resultados e conclusões	De acordo com o autor o resultado da simulação de erosão hidráulica é razoável e a aplicação carece na disponibilidade de entrar com dados de escolha do usuário. O autor conclui dizendo que com adição de novos efeitos visuais e algoritmos mais completos o programa mostra potencial.

Fonte: elaborado pelo autor.

No Craftscape a construção do terreno é feita através de uma malha hexagonal e suas elevações são calculadas através de um algoritmo gerador de ruídos (BOESCH, 2011). A velocidade e fluxo da água são calculadas através de um algoritmo baseado na diferença de altura do terreno. A erosão acarreta primeiramente na conversão da rocha para solo e no deslocamento de solo no sentido do fluxo (BOESCH, 2011). A Figura 6 demonstra a mudança do solo após aplicar erosão hidráulica na aplicação Craftscape (2011).

Figura 6 – Mudança de solo no Craftscape



Fonte: elaborado pelo autor.

3 DESCRIÇÃO

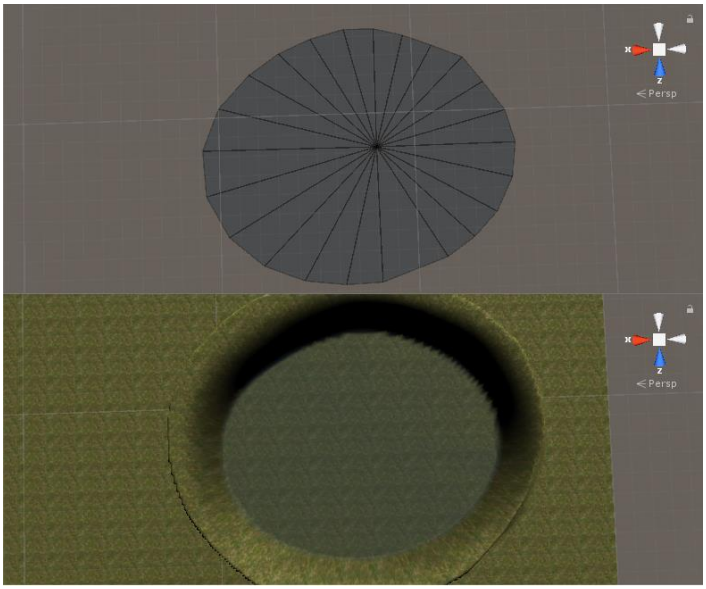
Neste capítulo é descrito o que foi desenvolvido e uma visão geral da aplicação. Na primeira seção é destacado as principais técnicas implementadas para a construção de funcionalidades diretamente relacionadas com o editor e técnicas usadas na garantia de objetivos que simulassem ambientes do mundo real. A segunda seção é apresentado uma visão geral do editor contemplando as telas e suas formas de uso.

3.1 IMPLEMENTAÇÃO

A implementação do editor foi feita na ferramenta Unity em conjunto com o Visual Studio 2017 utilizando a linguagem de programação C#. O posicionamento dos objetos, que englobam tipos de vegetação e edifício, é feito a partir do uso da `API Physics.Raycast`. Para o funcionamento do `Raycast` é necessário definir um ponto de origem, uma condição padrão de parada podendo ser distância máxima ou colisão e a direção do raio em relação a origem. No caso desta funcionalidade a origem é o cursor, o destino é a colisão com o terreno e a direção é da tela para o espaço do mundo. Por fim se obtém a posição do ponto destino que se atribui a posição do objeto que se deseja posicionar no terreno. Este método de posicionamento no terreno é usado em todos os objetos que podem ser colocados no terreno.

Para preencher locais do terreno com água foi implementado a construção de malhas triangulares via código fonte, utilizando os componentes `MeshFilter`, `MeshRenderer` e auxílio do `Raycast`. Primeiramente se utiliza o `Raycast` para escolher o local onde será preenchido com água e este mesmo ponto se torna o primeiro vértice e é localizado no centro da malha em relação aos demais vértices. Para identificar a área que será preenchida e obter os demais vértices se utiliza o `Raycast` com a origem sendo o primeiro vértice da malha, o parâmetro de direção sendo o eixo Z do objeto e o destino a colisão com algum objeto do terreno ou o próprio terreno. Para obter os demais vértices, que delimitam o tamanho da malha, é aplicado uma rotação do objeto em um ângulo calculado com base na quantidade de vértices que a malha irá conter, ou seja, quanto mais vértices menor o ângulo entre cada uso do `Raycast`. Com a informação do primeiro vértice e dos demais pontos obtidos pelo `Raycast` é feito a construção da malha. Os triângulos são constituídos com o primeiro vértice, o central, e outros dois obtidos pelo uso do `Raycast` no eixo Z, onde devem ser vértices vizinhos. Para limitar o terreno foi posicionado paredes invisíveis ao redor no terreno para a etapa do `Raycast` retornar um valor válido e completar o processo de geração da malha. A Figura 7 apresenta no primeiro momento a malha de água fora do terreno com destaque na disposição dos triângulos. E no segundo momento a Figura 7 mostra o terreno em que esta malha foi criada. A implementação desta técnica encontra-se no Apêndice A.

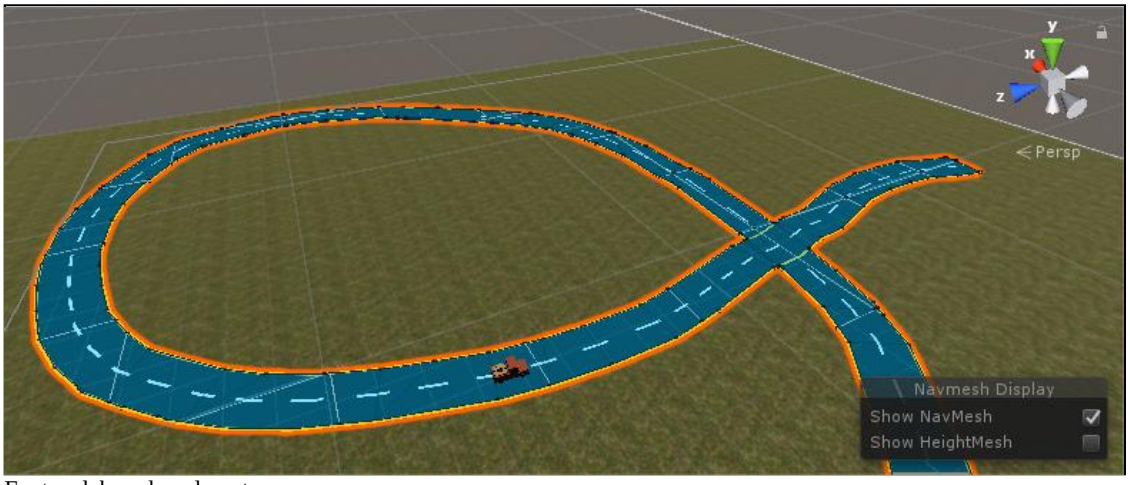
Figura 7 – Malha de água construída via script



Fonte: elaborado pelo autor.

Para construir estradas foi feito o uso do componente `TrailRenderer`. A posição do objeto que contém o `TrailRenderer` atribuído acompanha a posição do cursor utilizando `API Physics.Raycast`. Com o objetivo de corrigir o rastro sempre de face para a câmera, ao entrar no modo de construção de estrada a câmera é reposicionada no topo do terreno, resultando em ruas regulares no valor da sua largura. Para transformar o rastro do `TrailRenderer` em uma malha do tipo `Mesh` é chamado o método público `BakeMesh()`, da interface de script referente ao componente `TrailRenderer`. Desta forma o rastro do `TrailRenderer` passa a ser uma malha padrão do Unity, porém em uma altura incorreta. Sendo assim, os vértices da malha recém-criada são iterados, alterando o valor de Y para o Y do terreno, para que o pavimento acompanhe as ondulações do terreno. É possível posicionar um carro sobre os pavimentos, pois foi implementado o sistema de navegação utilizando os componentes `NavMeshSurface` e `NavMeshAgent` do pacote `NavMesh`. São sorteados pontos de destino na malha de navegação para que o veículo ande livremente sobre o pavimento. Cada vez que o veículo chega ao destino um novo ponto é sorteado. Na Figura 8 pode-se observar a estrada com aplicação do `NavMesh Surface` em cor azul.

Figura 8 – Aplicação do `NavMesh Surface` na estrada

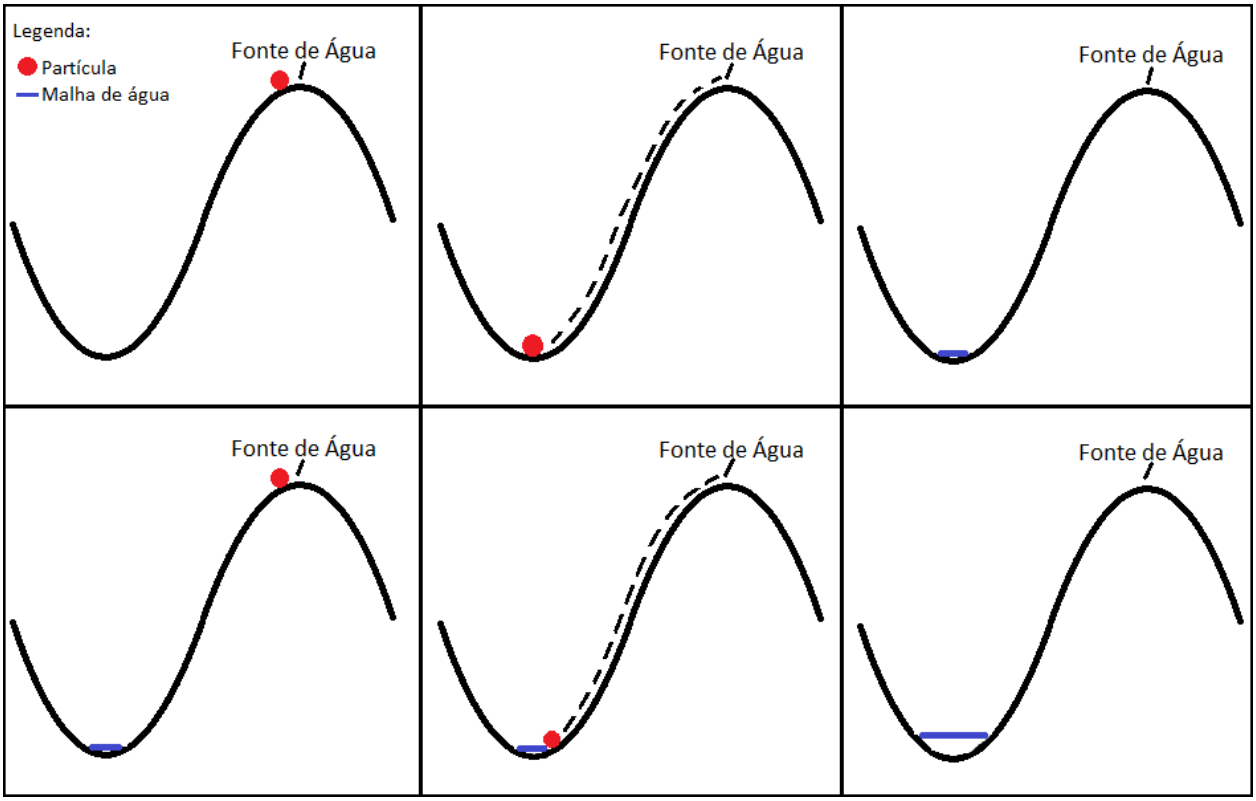


Fonte: elaborado pelo autor.

Além da implementação de funcionalidades de disposição de objetos pelo terreno, foi desenvolvido simulações de chuva, nascente e evaporação por meio de um regulador de temperatura. Tanto a chuva e a nascente foram construídas utilizando `ParticleSystem`. O sistema de partículas é um conjunto de módulos configuráveis que modificam o modo de apresentação das partículas. As partículas são objetos que possuem tempo de vida e são instanciadas em larga escala. Dos 22 módulos disponíveis, apenas 5 são utilizados, são eles, `Emission`, `Shape` e `Renderer` que são obrigatórios para que haja partículas, e `Collision` e `Trails`, um para tratar colisões e outro para criar rastros do fluxo da água para cada partícula.

Nesta aplicação se usa as configurações de física padrão, ou seja, as partículas não têm nenhum script de alteração de velocidade com exceção da configuração da velocidade inicial, necessária para que ela se mova após ser gerada. Mesmo que visivelmente a partícula esteja parada se usa uma margem no teste da condição para perceber se a partícula não se move, pois os objetos que estão sobre o efeito da física dificilmente têm o valor da velocidade em zero absoluto. Caso identifique que uma partícula está parada de acordo com a condição implementada, seu tempo de vida é colocado em zero e na última coordenada que a partícula esteve é criado a malha de água descrita anteriormente. Em inundações quando a partícula entra em contato com a água é chamado o método de atualização da malha de água, aumentando seu valor do eixo Y e recalculando as margens da malha. Na Figura 9 são apresentados os momentos de criação de malha por meio de uma partícula e após isto o recálculo feito quando a partícula colide com a região hídrica causando o aumento no nível da água.

Figura 9 – Atualização da malha de água em relação ao valor Y



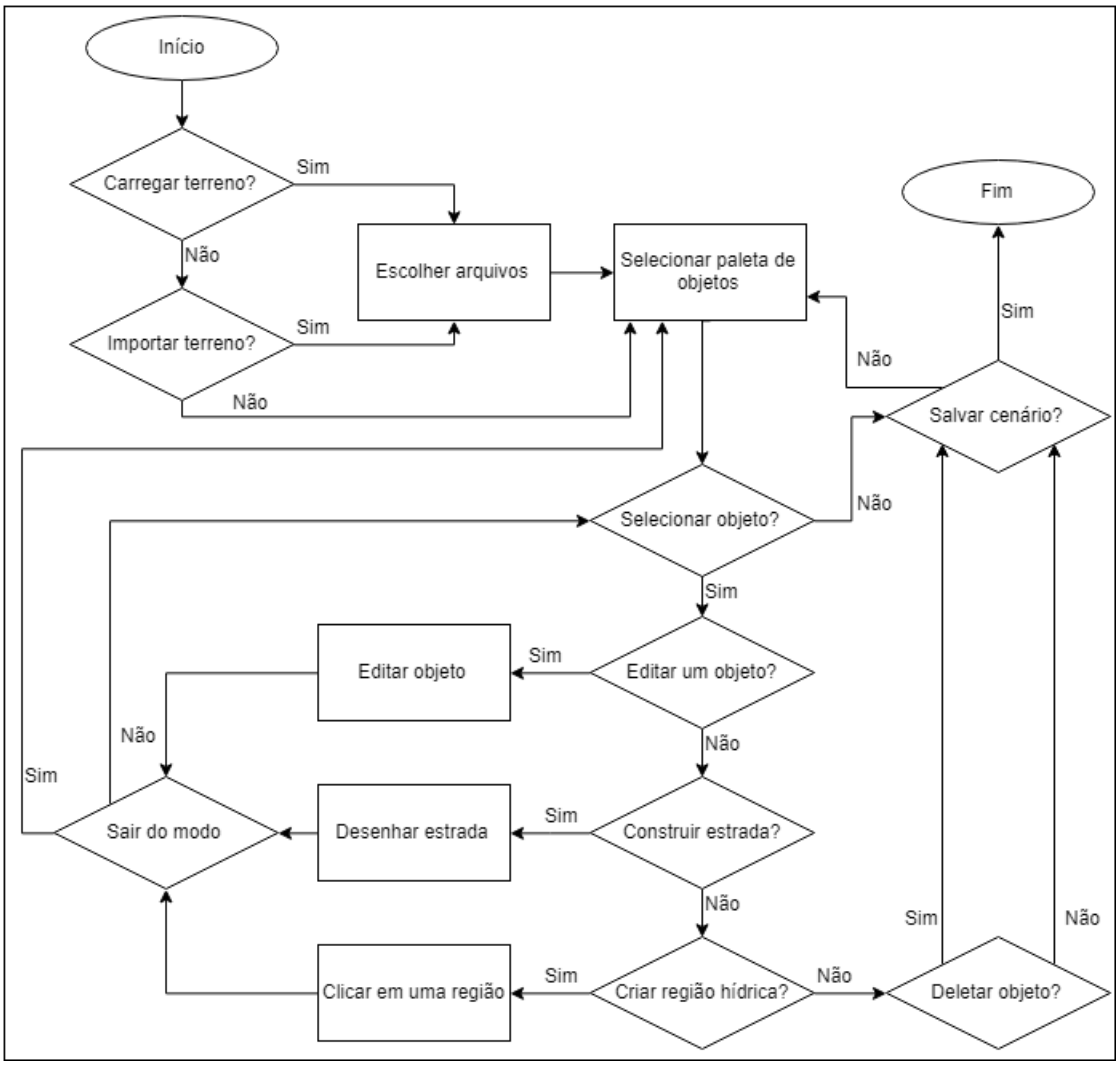
Fonte: elaborado pelo autor.

3.2 VISÃO GERAL DO EDITOR

O editor desenvolvido mantém algumas características do editor de Age of Empires (1999). Os principais atributos do editor que pertencem a tela principal envolvem uma paleta de objetos, um painel de configuração do objeto, um menu e a exibição do terreno que contempla a maior parte da tela. Além da tela principal, é possível acessar um menu. Na Figura 10 é visualizado o fluxograma da aplicação e na Figura 11 é observado a interface principal do editor.

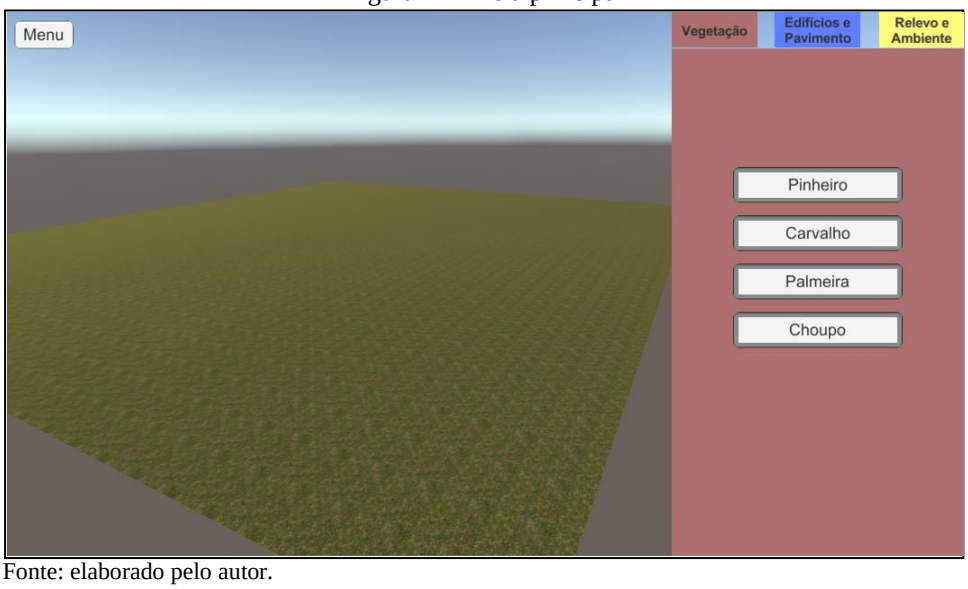
No canto direito da tela encontra-se a paleta de objetos que podem ser postos no terreno. Ela é dividida em três categoria: a primeira `Vegetação`, a segunda `Edifícios e Pavimento` e a terceira `Relevo e Ambiente`. Encontra-se na categoria `Vegetação` quatro tipos de árvore pinheiro, carvalho, palmeira e choupo. Na segunda categoria, dos `Edifícios e Pavimento`, há quatro tipos de edifícios que são academia, casa azul, prédio amarelo e prédio comercial. Além disto tem o modo de construção de estradas e um botão para adição de veículos. Na última aba chamada `Relevo e Ambiente`, contém um regulador de temperatura, botões para posicionamento de nascentes ou nuvens, e um botão para o modo de geração de locais hídricos. A Figura 12 mostra a interface de cada aba descrita.

Figura 10 – Fluxograma do editor



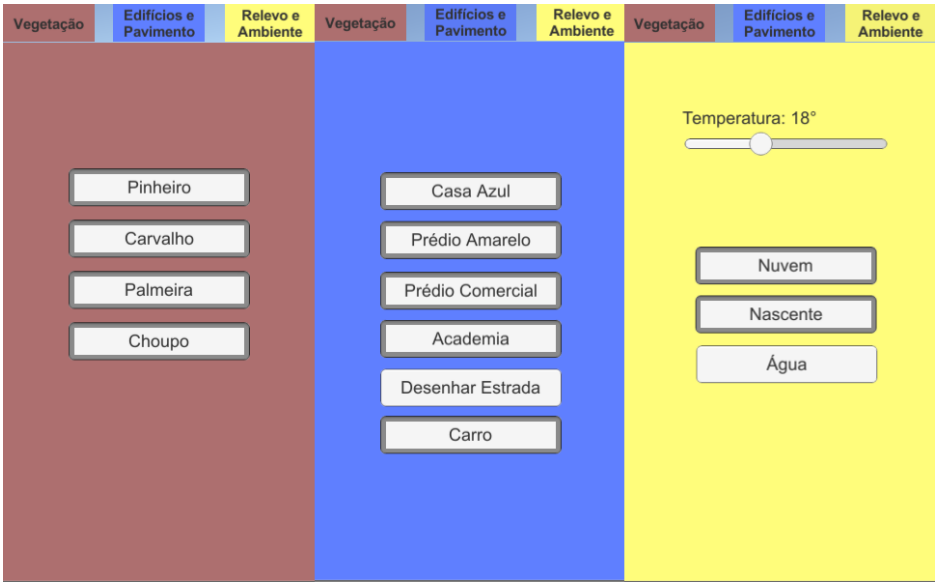
Fonte: elaborado pelo autor.

Figura 11 – Tela principal



Fonte: elaborado pelo autor.

Figura 12 – Paleta de objetos aberta



Fonte: elaborado pelo autor.

Após selecionar um objeto da paleta e arrastando-o para o terreno fica visível o painel de configuração do objeto atualmente selecionado. Dependendo do objeto o painel possui particularidades. Se for selecionado objetos do tipo árvore ou edificações o painel contém o nome do objeto, a imagem, posição em coordenadas X e Y, um regulador de rotação e escala. Se o objeto for uma nuvem ou nascente o objeto contém ao invés de um regulador de escala um botão para ligar e outra para desligar a fonte de água. Se for uma estrada ou uma região hídrica apenas mostra o nome e imagem deste tipo de conteúdo. Caso se opte por reconfigurar um objeto já posto no terreno basta clicar com o botão esquerdo do mouse sobre o objeto, sendo possível após selecionado o objeto excluí-lo clicando na tecla DEL no teclado. E se for necessário reposicionar um objeto já em cena deve-se pressionar o botão esquerdo do mouse sobre o objeto por 1s e após este tempo mover o mouse, não sendo permitido posicionar um objeto sobre o outro. Nestas circunstâncias o objeto selecionado fica com a cor avermelhada. A Figura 13 mostra as variações dos painéis de configuração.

Figura 13 – Painéis de configuração

Fonte: elaborado pelo autor.

Para navegar no cenário editado foi implementado um controle de câmera. Através das teclas W, A, S e D o movimento é para frente, para esquerda, para trás e para direita respectivamente. Ao pressionar a tecla SHIFT esquerda é possível rotacionar a câmera movimentando o mouse. Algumas mensagens são visualizadas no canto inferior da tela, após entrar em algum modo de edição é notificado como sair de um modo. E por fim, pode-se acessar um menu através do botão `Menu`. O botão pode ser visualizado no canto esquerdo superior na Figura 11.

O menu é composto por seis botões, como pode ser visto na Figura 14. O botão `Voltar` apenas retorna à edição atual. O botão `Novo` reinicia um terreno do zero. O botão `Salvar` salva o terreno e os objetos dispostos na atual edição. O botão `Carregar` carrega o terreno e os objetos sendo necessário escolher o arquivo que deseja ser carregado. O botão `Importar` importa a altura de um terreno, sendo necessário selecionar uma imagem de mapa de altura em escala de cinza. E o botão `Sair da Aplicação`, fecha o editor.

Figura 14 – Menu

Fonte: elaborado pelo autor.

4 RESULTADOS

Para a validação das funcionalidades, técnicas e simulações foi criado cenários de testes. Para isto foram construídos cenários controlados e não controlados, onde algumas imagens apresentam mais de uma funcionalidade. O Quadro 5 faz a relação entre características e cenários, onde as figuras que possui certo tipo de característica estão marcadas com um "X".

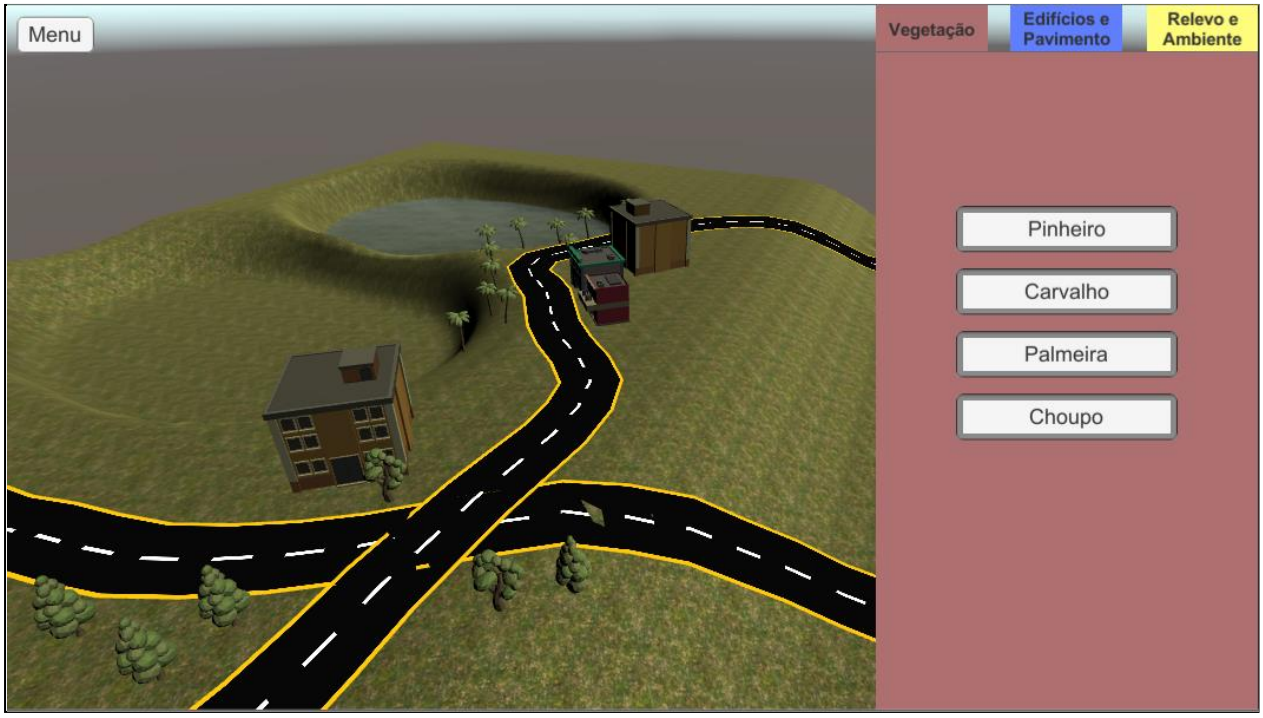
Quadro 5 – Relação entre características e cenários

	Cenários controlados				Cenários não controlados			
Cenários / Características	Figura 15	Figura 16	Figura 17	Figura 18	Figura 19	Figura 20	Figura 21	Figura 22
Vegetação	X	X	X				X	X
Edificações	X	X	X				X	X
Região hídrica	X	X	X	X	X	X		X
Estrada	X	X	X				X	X
Estradas com cruzamentos	X						X	
Simulação chuva				X		X		
Simulação nascente		X	X		X			
Simulação evaporação				X				
Simulação inundação				X	X	X		
Fluxo da água		X	X	X	X	X		

Fonte: elaborado pelo autor.

Na Figura 15 é apresentado a funcionalidade de posicionamento de objetos utilizando `API Physics.Raycast`, criação da região hídrica com construção do `Mesh` via script e construção de estrada com cruzamento utilizando componente `TrailRenderer`.

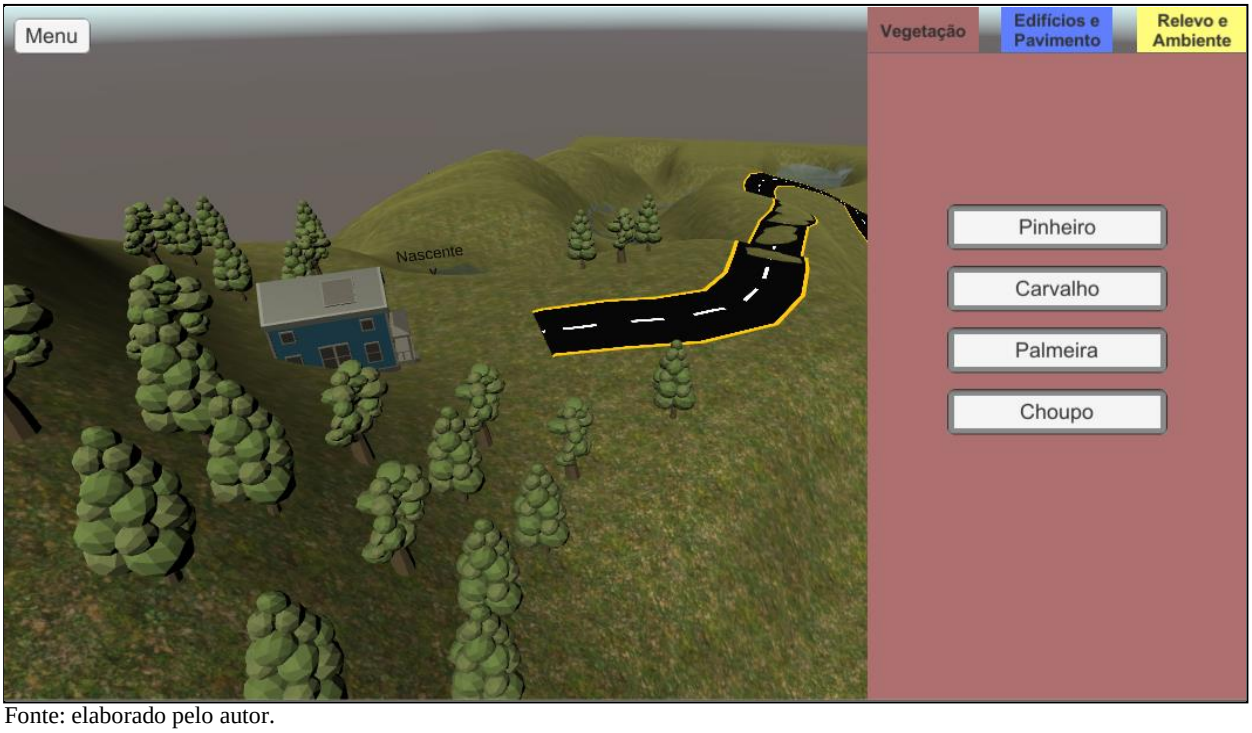
Figura 15 – Cenário controlado 1



Fonte: elaborado pelo autor.

A Figura 16 apresenta posicionamento de vegetação e edificações em regiões montanhosas, a construção de estradas em terrenos contendo elevações, posicionado uma nascente gerando um fluxo de água através do `Particle System` e ao fundo no canto direito superior uma região hídrica.

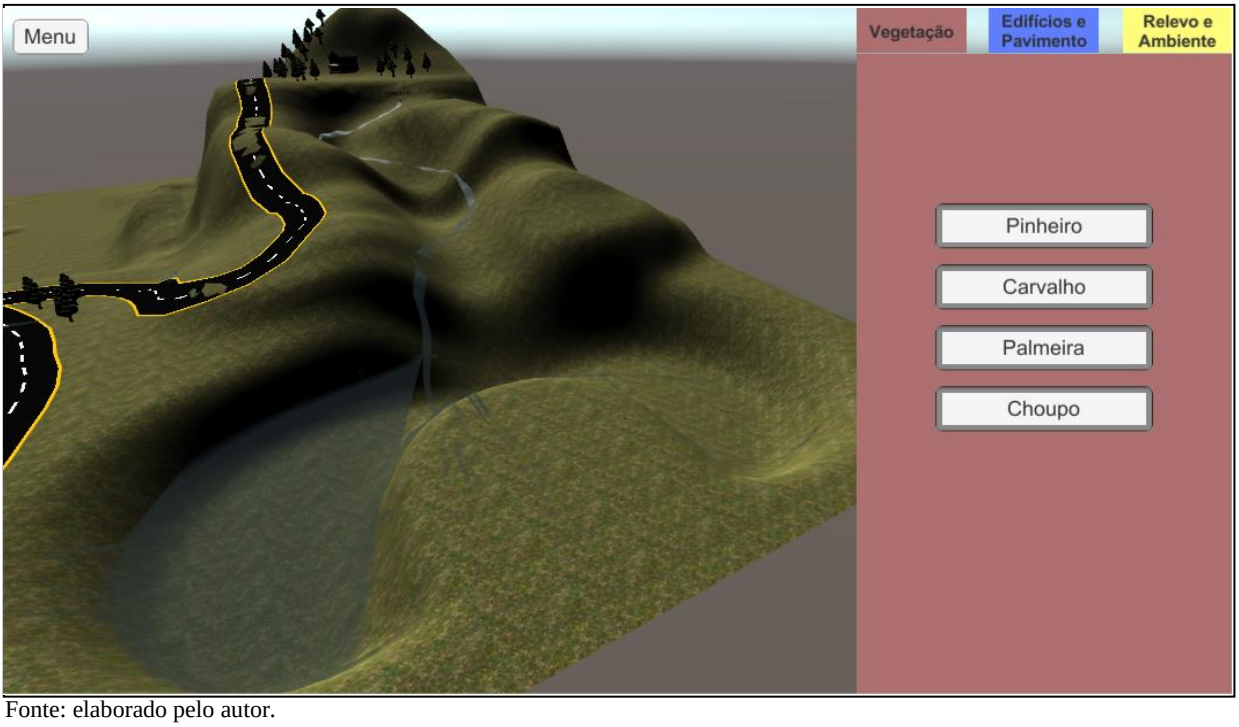
Figura 16– Cenário controlado 2



Fonte: elaborado pelo autor.

A Figura 17 é o mesmo ambiente da Figura 16, porém com outro ângulo para destaque no fluxo da água, e na criação da água através da partícula em uma depressão irregular.

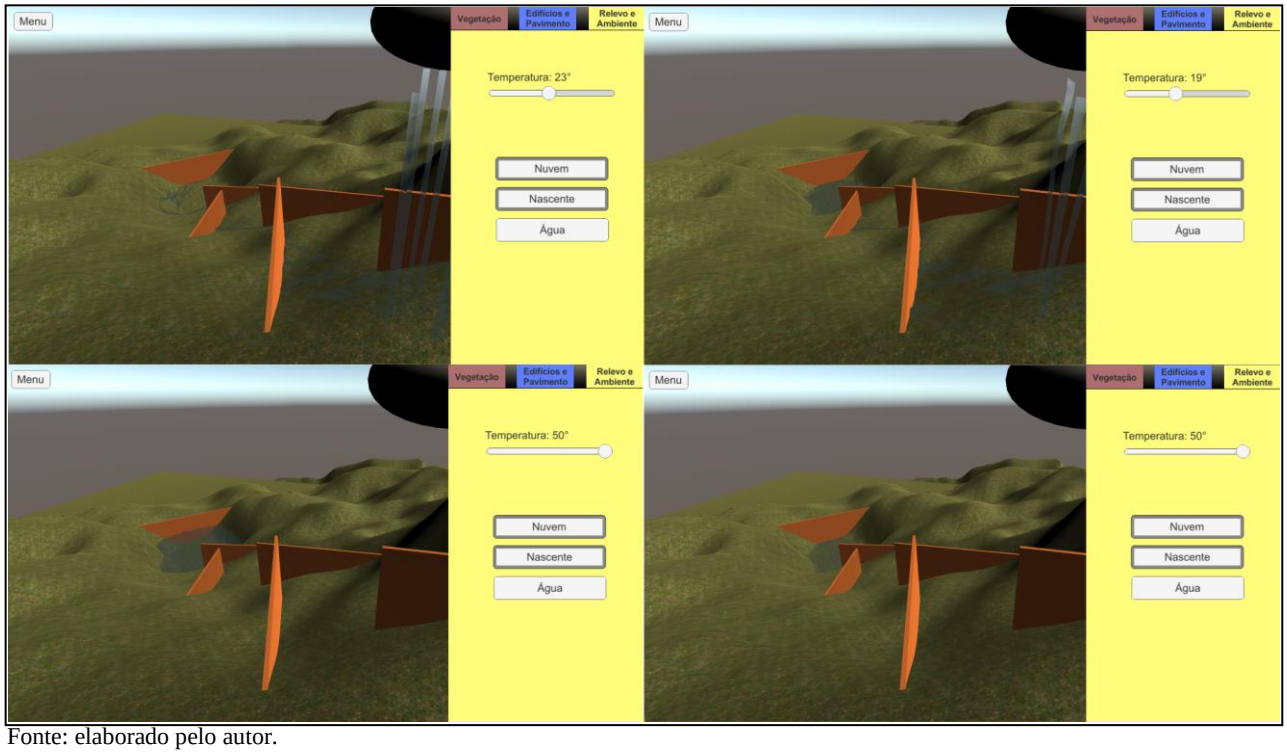
Figura 17 – Cenário controlado 3



Fonte: elaborado pelo autor.

A Figura 18 são etapas de um mesmo cenário mostrando o fluxo da água, criado por uma nuvem, em relação aos obstáculos laranjas, a inundação ocorrida mais ao fundo e a evaporação após desligar o fluxo causado pela chuva e a temperatura ser regulada para 50º.

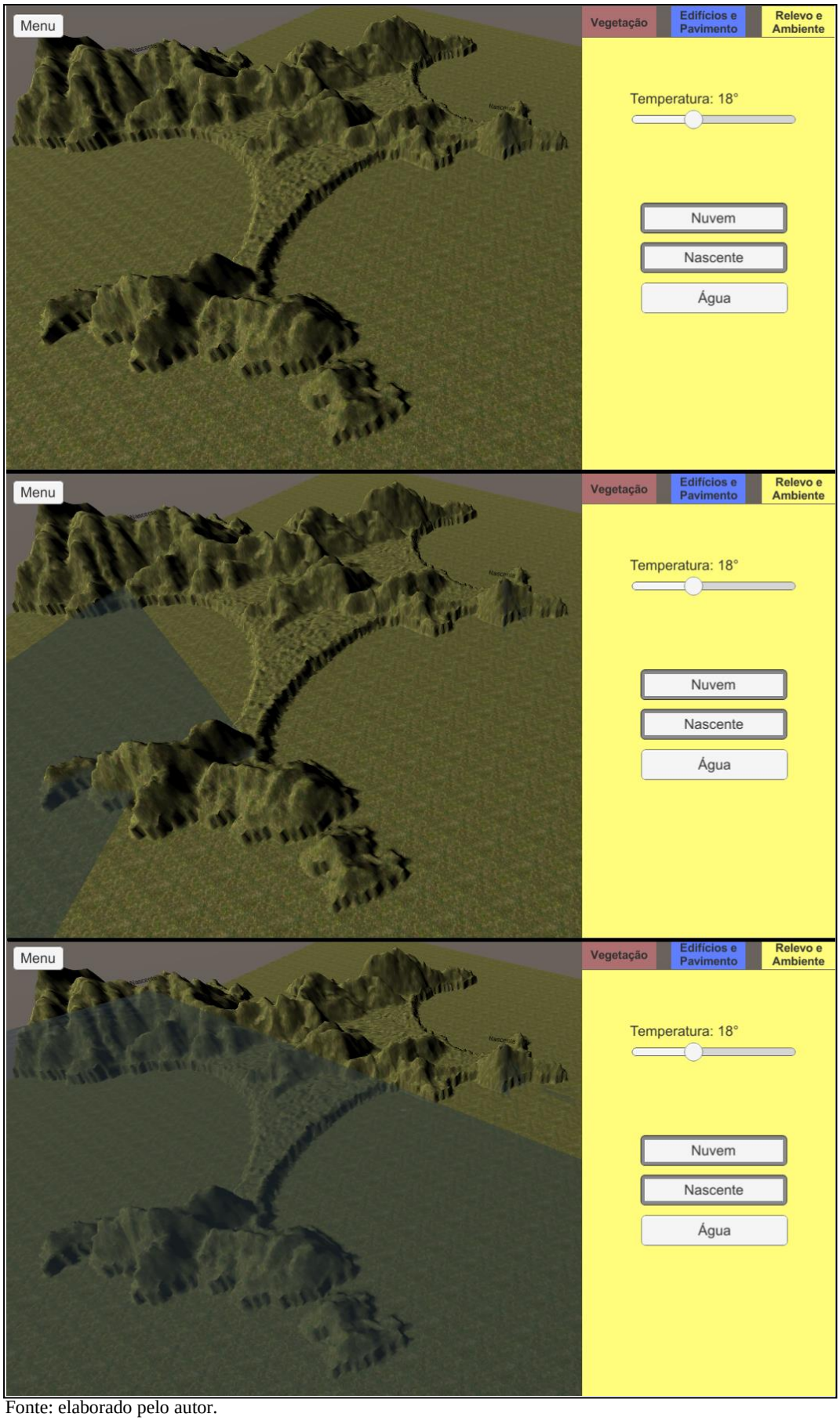
Figura 18 – Cenário controlado 4



Fonte: elaborado pelo autor.

Figura 19 são três estágios após ligar o fluxo de água em duas nascentes localizadas em dois morros da região. A malha de água gerada foi em função da implementação de monitoramento de parada da partícula.

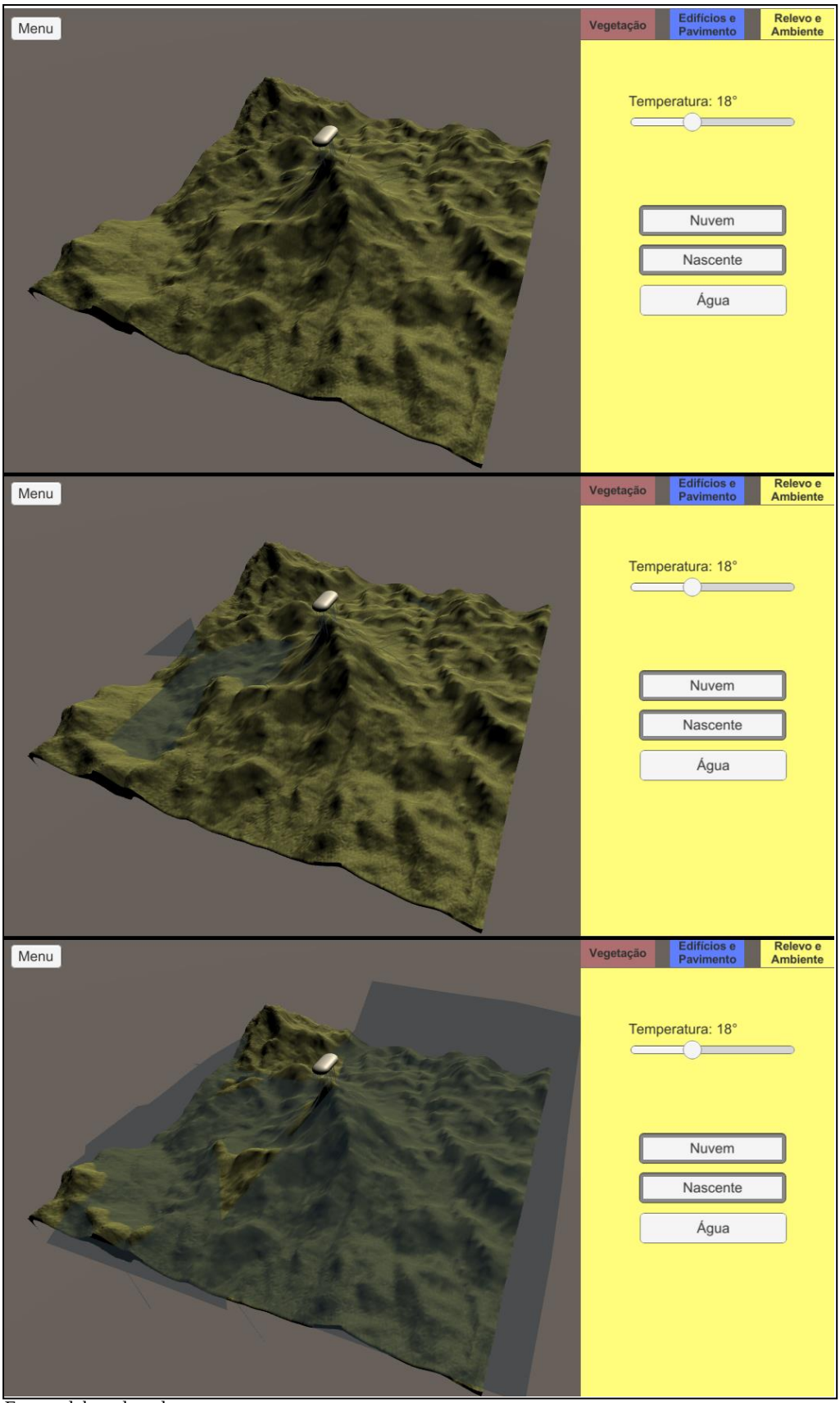
Figura 19 – Cenário não controlado 1



Fonte: elaborado pelo autor.

Na Figura 20 foi posicionada uma nuvem no topo do terreno e ligado o fluxo de chuva. Desta forma a imagem marca momentos da reação das partículas em relação ao terreno e física resultando em inundações pelo terreno.

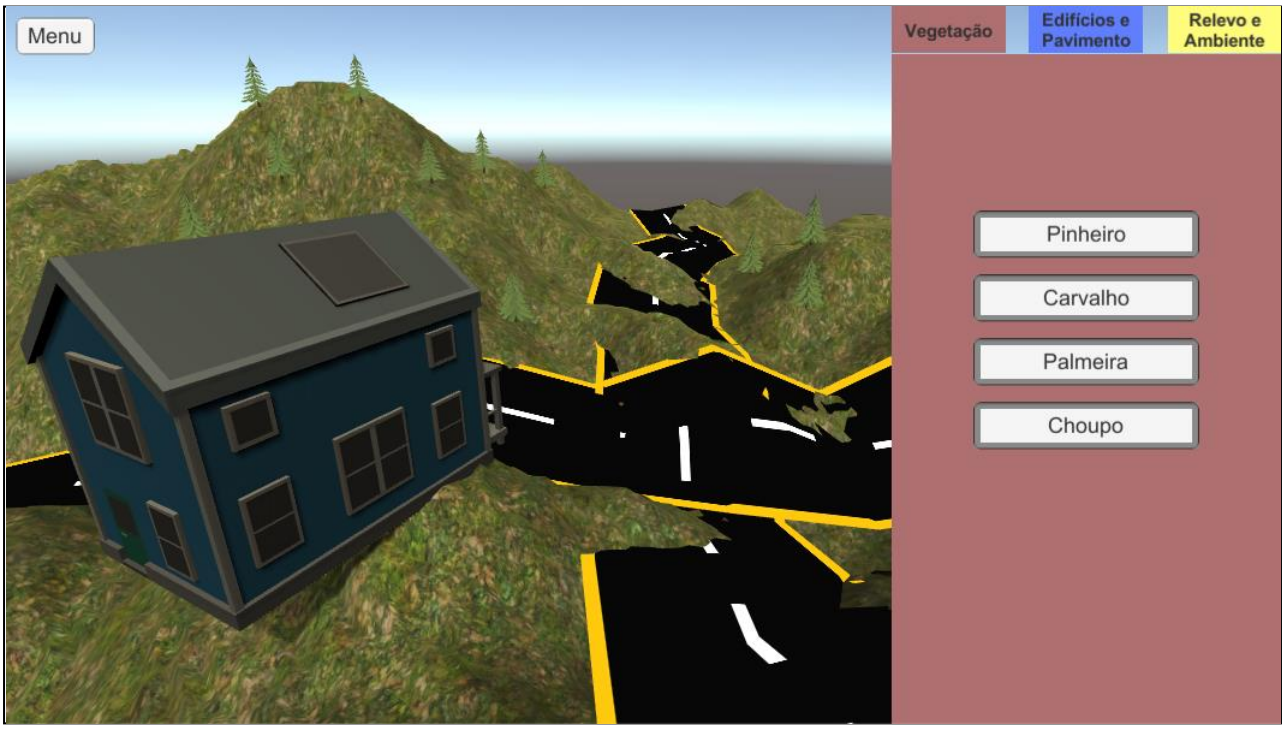
Figura 20 – Cenário não controlado 2



Fonte: elaborado pelo autor.

A Figura 21 mostra a disposição de objetos em um ambiente não controlado em um terreno irregular juntamente a estrada com cruzamento na via. É observado neste teste a sombra gerada pela base da casa no solo, pelo fato do terreno não ser aterrado conforme a disposição do objeto casa.

Figura 21 – Cenário não controlado 3



Fonte: elaborado pelo autor.

Figura 22 apresenta a criação de uma região hídrica, disposição de objetos e a construção de uma estrada. Novamente destaque para o prédio que está sobre um morro e a estrada que mesmo sendo construída em um terreno não controlado, se comparado a Figura 21, se comportou melhor em relação a irregularidade do terreno.

Figura 22 – Cenário não controlado 4

Fonte: elaborado pelo autor.

Da Figura 15 até Figura 18 foram utilizados terrenos modelados através das ferramentas disponibilizadas pelo ambiente Unity. Os cenários não controlados, representados pela Figura 19 a Figura 22, foram importados através da

leitura de mapas de altura capturados pelo correlato terrain.party (2015), sendo possível visualizá-los no Anexo A. Estes mapas tinham a resolução original de 1081x1081 e foram redimensionados para 512x512, pelo fato do tamanho do terreno no Unity ser 512x512. A Figura 19 representa a região do município de Bombinhas, no estado de Santa Catarina e da Figura 20 a Figura 22 está representado o Morro do Cachorro, localizado em Blumenau também localizado no estado de Santa Catarina. A maior irregularidade ocorre em regiões litorâneas, que é o caso da Figura 19, onde resulta em uma região exageradamente íngreme. Portanto através da leitura de um mapa de altura é possível representar um terreno do mundo real no terreno deste editor respeitando os formatos, mas não a escala.

O posicionamento dos objetos em terrenos planos foi bem-sucedido, porém em solos irregulares apresentou falhas visuais. As edificações quando postas em locais íngremes ou nos picos dos morros não simulam a física do mundo real. A base destes objetos fica visível ou dentro do terreno. Visualmente isto não ocorre com as árvores por suas bases serem menores. Estes casos podem ser analisados na Figura 15, Figura 16, Figura 17, Figura 21 e Figura 22. No editor de Age of Empires II a perspectiva é limitada em 4 ângulos, e unidades e construções possuem uma liberdade no seu posicionamento proibindo apenas um objeto sobre o outro. Entretanto em casos de posicionamento de elementos temáticos de cenário o terreno se transforma em uma espécie de tabuleiro limitando o posicionamento destes elementos a apenas posições específicas.

A disposição de regiões hídricas é necessária uma análise caso a caso. O objetivo principal desta implementação era modelar a malha de água de acordo a percepção do terreno encontrado no local da sua posição. Este exemplo é apresentado na Figura 15, onde encontra-se duas depressões bem próximas uma da outra e enquanto uma está com a área totalmente preenchida a outra está vazia. Entretanto, na Figura 17 é apresentado um caso de que a técnica não cobre toda a área de uma mesma depressão, pois o formato do desnível não permite, resultando em apenas metade coberto de água. Este problema tem haver em alguns casos com a localização do vértice central da malha da água, podendo examinar na Figura 19, Figura 20 e Figura 22 onde a malha fica com um formato irregular e não atinge toda a margem de um desnível. No editor do jogo da Ensemble Studios (1999), as regiões hídricas apenas são texturas sobre o terreno com o mínimo de animação a nível 2D. Na aplicação de Boesch (2011), encontra-se uma simulação mais próxima do real.

A construção de estradas foi possível ser feita tanto em terrenos planos quanto em irregulares. Como a sua criação é através do mouse, ocorre que a malha do pavimento não fica completamente linear. Em elevações suavizadas, que é o caso da Figura 15 e Figura 16, a estrada apresenta trechos sem defeitos e sem cortes com o terreno. Em cenários como na Figura 17, Figura 21 e Figura 22, as irregularidades ficam mais visíveis. Conforme implementado, os vértices da malha respeitam a altura do terreno, mas em casos de inclinações do solo a visualização da estrada não condiz com o que ocorre no mundo real. Em situações de cruzamento da via, Figura 15 e Figura 21, a visualização apresenta falhas pelo fato da textura da malha estar dividindo o mesmo espaço de forma que resulte em uma confusão na renderização da Unity. Destaca-se que as estradas estão ligadas com a criação das malhas de navegação, que não são visíveis. Ao criar um cruzamento na via a malha de navegação é atualizada e os veículos passam a utilizar o cruzamento nos cálculos de distância. Pode-se observar estradas no editor de Age of Empires (1999), porém é apenas uma textura do terreno ao contrário da criação de malha que ocorre desconectada do terreno. Leva-se em conta também que o tema do jogo é batalhas medievais de forma que o jogo não contempla estradas com texturas iguais a apresentada neste trabalho.

A simulação da chuva visualmente não remete ao mundo real, podendo ser visto na Figura 17 e Figura 18. A nascente, visualizada na Figura 16, Figura 17 e Figura 19, é representada com um marcador 2D. O módulo `Trail` dos sistemas de partículas juntamente a física padrão do Unity criam um fluxo de água, porém usou-se um número baixo na emissão de partículas resultando em um fluxo estreito, como pode ser observado da Figura 16 até a Figura 20. A inundação ocorre na Figura 18, Figura 19 e Figura 20, mas com falhas. Em ambientes controlados, que é o caso da Figura 18, a inundação ocorre normalmente até transbordar. No momento de transbordar a técnica de atualização da malha apenas procura a margem mais próxima ao invés de voltar ao estado de procura de uma região mais baixa gerando um novo fluxo. Em cenários não controlados as imperfeições do terreno e a aleatoriedade de reação da partícula em relação a física e o terreno causa inundações errôneas gerando malhas de água com formatos irregulares, sendo o caso da Figura 19 e Figura 20. Alguns casos a partícula encontra o lugar mais próximo, mas o problema fica no local que a partícula parou. O local de parada vira o vértice central da malha de água, resultando no mesmo erro de localidade do vértice descrito anteriormente. Em outros casos a partícula que está sobre efeito da física pode ganhar muita velocidade e usar uma região do terreno como rampa fazendo com a partícula suba perpendicularmente para o alto em relação ao chão. A gravidade fará com que ela volte e quando atingir velocidade zero antes de começar a cair, vai ser criado uma malha na região errada. Esta mesma situação ocorre quando uma partícula fica em uma depressão côncava de um lado para o outro.

A evaporação é apresentada na Figura 18, onde é destacado o ajuste da temperatura de 19º para 50º e o desligamento da chuva. A região hídrica vai baixando seu nível até sumir sem atualizar o tamanho da malha. Após isto o objeto da malha é excluído da cena por não ser mais visível. A evaporação é resultado de uma inundação correta, pois se a região for mal coberta pela malha da água visualmente a evaporação não irá corresponder de forma positiva. Nesta aplicação há a representação visual da fonte de água em nascentes e nuvem, característica que não aparece na aplicação

de Boesch (2011). Porém a simulação no fluxo da água e a inundação encontrada em Boesch (2011) remete mais ao mundo real do que a implementada neste trabalho.

5 CONCLUSÕES

O presente trabalho busca criar um ambiente capaz de ser editado de acordo com o interesse do usuário que busca visualizar fenômenos naturais e artificiais através do ambiente virtual. O objetivo é alcançado quando se deparado com cenários controlados com terrenos prontamente editados externamente, falhando ainda em alguns aspectos como posicionamento de objeto com uma base grande, havendo a necessidade de corte no terreno abaixo do objeto. A aplicação permite importar terrenos do mundo real através de mapas de altura obtidos por ferramentas como terrain.party (2015). Os formatos da região foram mantidos, porém para a leitura correta de toda a imagem é necessário regular sua resolução. Com o ajuste na resolução o terreno importado perde o fator de escala em relação aos objetos disponibilizados para uso.

A criação da malha hídrica em depressões de formato regular atinge o objetivo de não inundar outras possíveis depressões mais próximas que devem ficar secas. Porém, em depressões com formatos irregulares ou possuindo porções de terreno dentro da área depressiva prejudicam a performance da técnica, necessitando uma otimização no código para estes casos. Uma recomendação seria utilizar a técnica desenvolvida e aprimorada na criação de malhas somente em cenários estáticos. Para simulações de fluídos e fluxos de algum tipo de líquido utilizaria apenas o sistema de partículas, pois foi pesquisado, durante do desenvolvimento, o uso das partículas na criação de fluídos em ambientes controlados e que respeitavam as físicas do mundo real.

O uso de partículas com `Trails` e a ação da física padrão do Unity alcançou o seu objetivo de criar os fluxos da água possibilitando observar o caminho que a água supostamente faria no mundo real. Porém ao usar as partículas juntamente a técnica de criação de malhas para gerar regiões inundadas somente apontou resultados expressivos em ambientes controlados. É necessária uma nova técnica no monitoramento de parada das partículas ou uma reconstrução completa no sistema de fluxo hídrico, como já citado anteriormente.

A técnica de construção de estradas também atingiu seu objetivo por explorar o componente `TrailRenderer`. Desta forma construir uma estrada é mais rápido do que utilizar modelos prontos e usá-los com a técnica de posicionamento via `API Physics.Raycast`. Em terrenos irregulares a textura do terreno fica visível, porém respeita a questão que não se constrói estradas em terrenos desnivelados, sugerindo a futura implementação de um planificador de região ou uma planificação automática no momento de construção do pavimento. Os cruzamentos da estrada não prejudicaram a criação da malha de navegação alcançando o objetivo de locomoção de veículos, porém sua representação não foi corrigida.

Os correlatos apresentados anteriormente no Quadro 2, Quadro 3 e Quadro 4 tem relações diretas e indiretas com o presente trabalho. Terrain.party (2015) tem o objetivo de levar os relevos do mundo real para o virtual. Esta afirmativa se resume em uma das motivações do trabalho e está presente em todas as funcionalidades dele. O editor de cenário de Age of Empires II (1999) é a base do trabalho, pois tem impacto na definição de funcionalidades como posicionamento de objetos, disposição da interface, formas de apresentar diferenciação de relevo entre outras. A aplicação desenvolvida por Boesch (2011), motiva a não apresentar um cenário estático, e, portanto, simular certos ciclos do mundo real.

As técnicas desenvolvidas neste trabalho parcialmente cumprem com seus objetivos e melhorias são necessárias para o editor se tornar uma ferramenta didática na apresentação de ciclos naturais e representações do mundo real. Por este motivo o trabalho possui possíveis extensões, como:

a) otimização na técnica de *hole fill* para malhas de polígonos;
b) adicionar função de corte de terreno;
c) correção da textura de estradas em caso de cruzamentos;
d) criação de sentido de fluxo de veículos nas estradas;
e) enriquecer a paleta de objetos com a opção de importação de objetos;
f) melhorar o monitoramento da velocidade das partículas na verificação de parada;
g) criar funções de edição de terreno;
h) pesquisar a construção de fluídos através do sistema de partículas.

REFERÊNCIAS

BOESCH, Florian. **WebGL GPU landscaping and erosion**. Basel, Suíça, 2011. Disponível em: <http://codeflow.org/entries/2011/nov/10/webgl-gpu-landscaping-and-erosion/>. Acesso em: 03 nov. 2018.

FRANCISCO, Wagner de Cerqueria e. **O Relevo**. [2018?]. Disponível em: <https://escolakids.uol.com.br/o-relevo.htm>. Acesso em: 31 de out. 2018.

GLOVER, Jesse. **Learn and Understand Raycasting in Unity3D**. 2017. Disponível em: < https://gamedevacademy.org/learn-and-understand-raycasting-in-unity3d/>. Acesso em: 29 de mar. 2019.

PENA, Rodolfo F. Alves. **Relevo e Sociedade**. [2018?]. Disponível em: <https://mundoeducacao.bol.uol.com.br/geografia/relevo-sociedade.htm>. Acesso em: 01 de nov. de 2018.

QUECONCEITO. Conceito de Simulador. São Paulo. [2017?]. Disponível em: <https://queconceito.com.br/simulador>. Acesso em: 02 de nov. de 2018.

SANGREMAN, Anderson; FREITAS, Gustavo; COSTA, Ramon R. Modelagem de Terrenos Naturais Através de Malhas de Triângulos, In: PROCEEDING SERIES OF THE BRAZILIAN SOCIETY OF APPLIED AND COMPUTATIONAL MATHEMATICS, Vol. 1, N. 1, 2013. **Anais eletrônicos...** Rio de Janeiro. p. 1-6. Disponível em: <https://proceedings.sbmac.org.br/sbmac/article/viewFile/132/132>. Acesso em: 02 de nov. 2018.

UNITY. **Anatomy of a Mesh**. [S.l], 2019a. Disponível em <https://docs.unity3d.com/Manual/AnatomyofaMesh.html>. Acesso em: 28 mar. 2019.

UNITY. **Materials, Shaders & Textures**. [S.l], 2019b. Disponível em < https://docs.unity3d.com/Manual/Shaders.html>. Acesso em: 28 mar. 2019.

UNITY. **Mesh Renderer**. [S.l], 2019c. Disponível em <https://docs.unity3d.com/Manual/class-MeshRenderer.html>. Acesso em: 28 mar. 2019.

UNITY. **Trail Renderer**. [S.l], 2019d. Disponível em <https://docs.unity3d.com/Manual/class-TrailRenderer.htmll>. Acesso em: 02 jun. 2019.

UNITY. **Using the Mesh Class**. [S.l], 2019e. Disponível em <https://docs.unity3d.com/Manual/UsingtheMeshClass.html>. Acesso em: 28 mar. 2019.

APÊNDICE A – CÓDIGO FONTE DA CONSTRUÇÃO DA MALHA DE ÁGUA

Neste apêndice é apresentado o código fonte de criação e construção da malha de água. O Quadro 6 é referente ao método `CreateShape()`, método responsável pela percepção de margens e organização dos dados necessário para concepção da malha de água.

Quadro 6 – Código fonte do método `CreateShape()`

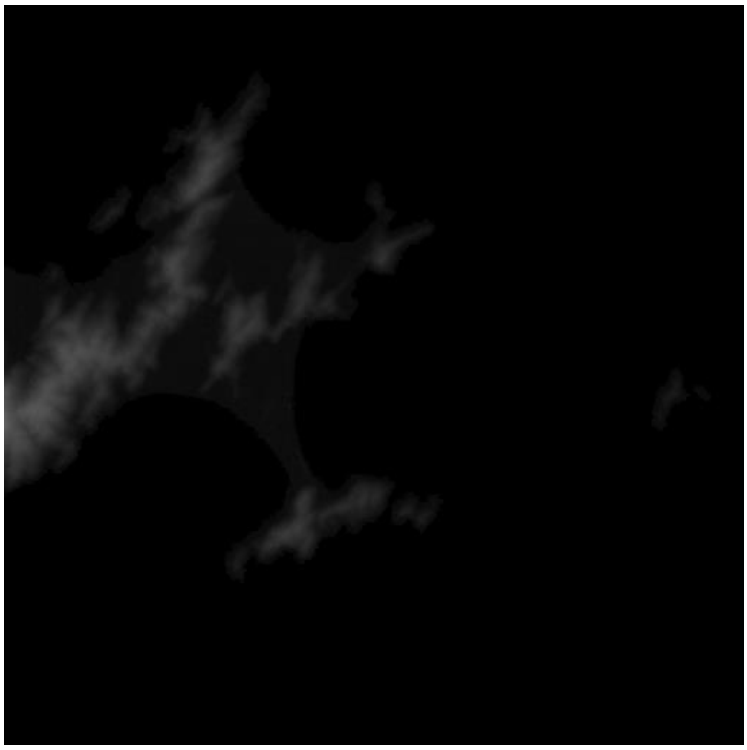
```
public void CreateShape()
{
  // Remoção do componente Collider para atualizar os vértices e no final adicionar um novo
   //Collider
  Collider[] childColliders = this.GetComponentsInChildren<Collider>();
  foreach (Collider collider in childColliders)
  {
      Destroy(collider);
  }

   // Inicialização do vetor de vértice e do primeiro vértice
   vertices = new Vector3[numVertices];

   // Coordenada do primeiro vetor em (0,0,0), respeitando a coordenada local do objeto
   vertices[0] = new Vector3(0, 0, 0);

   // Iteração para traçar os raios via Raycast e encontrar as margens
  for (int i = 1; i < numVertices; i++)
  {
        RaycastHit hit;

        if (Physics.Raycast(transform.position, transform.forward, out hit))
        {
            int posX = (int)(hit.point.x - transform.position.x);
            int posZ = (int)(hit.point.z - transform.position.z);
            // Quando encontrado a margem, vértice é criado e alocado no vetor
            vertices[i] = new Vector3(posX, 0, posZ);
        }
        else
        {
            Destroy(this.gameObject);
        }
        // Rotação do objeto para um novo disparo de Raycast
        transform.eulerAngles += new Vector3(0, (int)(360 / (numVertices - 1)), 0);
  }

   // Inicialização do vetor de triângulos
   triangles = new int[(numVertices - 1) * 3];
   int indiceTriangles = 1;

   // Iteração para adicionar os triângulos relacionando o índice do vetor de vértices
   // Exemplo, primeiro triângulo serão os três primeiros vértices do vetor de vértices
   // triangles[0] = 0
   // triangles[1] = 1
   // triangles[2] = 2
   // O segundo triângulo será o primeiro vértice e os alocados na posição 3 e posição 4
   // triangles[3] = 0
   // triangles[4] = 3
   // triangles[5] = 4
    for(int i = 1; i < numVertices; i++)
    {
        if (i == numVertices - 1)
        {
            triangles[indiceTriangles - 1] = 0;
            triangles[indiceTriangles] = i;
            triangles[indiceTriangles + 1] = numVertices - i;
        }
        else
        {
            triangles[indiceTriangles - 1] = 0;
            triangles[indiceTriangles] = i;
            triangles[indiceTriangles + 1] = i + 1;
        }
        // Índice somado em 3 por alocar de 3 em 3 posiçoes
        indiceTriangles += 3;
    }
}
```

Fonte: elaborado pelo autor.

ANEXO A – MAPAS DE ALTURA USADOS EM TESTES

Este anexo apresenta os mapas de altura utilizado em testes de cenários não controlados para a validação das funcionalidades desenvolvidas no trabalho. Na Figura 23 refere-se ao município de Bombinhas em Santa Catarina.

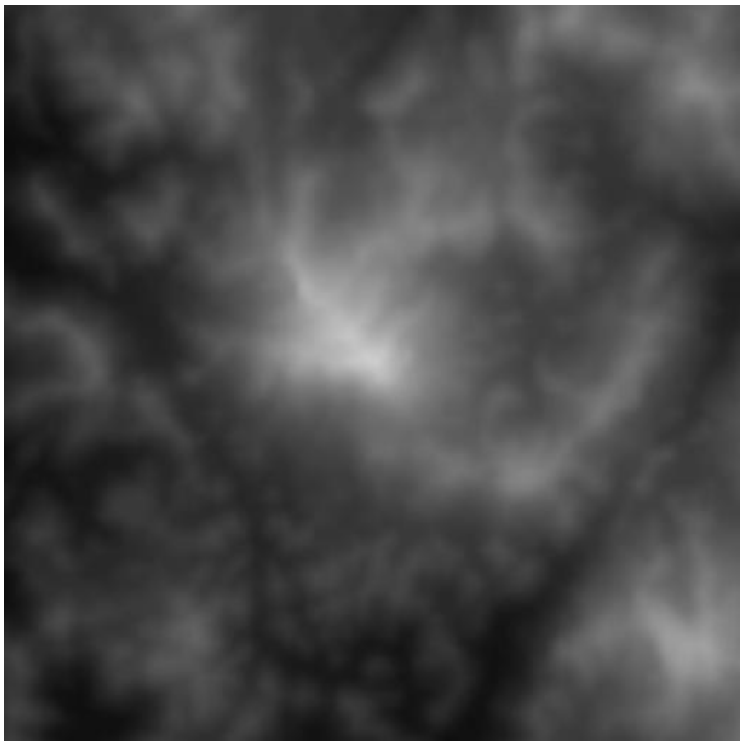
Figura 23 – Mapa de altura município de Bombinhas



Fonte: terrain.party (2019).

Na Figura 24 é apresentado o mapa de altura da região do Morro do Cachorro localizado no município de Blumenau em Santa Catarina.

Figura 24 – Mapa de altura do Morro do Cachorro



Fonte: terrain.party (2019).