UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Paulo Henrique Guimarães de Menezes

Geração Procedural de Conteúdo para Jogos de Realidade Aumentada

Recife, Brasil

Janeiro - 2018

Paulo Henrique Guimarães de Menezes

Geração Procedural de Conteúdo para Jogos de Realidade Aumentada

Trabalho de Conclusão de Curso apresentado ao Programa de Bacharelado em Ciência da Computação do Departamento de Estatística e Informática da Universidade Federal Rural de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Estatística e Informática

Programa de Graduação

Orientador: João Paulo Lima

Recife, Brasil

Janeiro - 2018

MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO (UFRPE)
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

http://www.bcc.ufrpe.br

FICHA DE APROVAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO

Trabalho defendido por Paulo Henrique Guimarães de Menezes às 16 horas do dia 08 de fevereiro de 2018, no Sala 31 do CEAGRI-02, como requisito para conclusão do curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, intitulado **Geração Procedural de Conteúdo para Jogos de Realidade Aumentada**, orientado por João Paulo Silva do Monte Lima e aprovado pela seguinte banca examinadora:

João Paulo Silva do Monte Lima
DEINFO/UFRPE

Filipe Rolim Cordeiro
DEINFO/UFRPE

Valmir Macário Filho
DEINFO/UFRPE

Agradecimentos

Primeiramente, agradeço aos meus pais, Eliane e Paulo por sempre me apoiarem nos momentos mais difíceis da graduação e sempre me incentivaram a nunca desistir dos meus objetivos.

Ao professor João Paulo Lima que me orientou no decorrer do último ano do curso. Obrigado pela orientação e pelo conhecimento transmitido tanto em suas disciplinas como no desenvolvimento desse trabalho.

Aos meus amigos que tenho desde o ensino médio, André Luiz e Lucas Apolinário por estarem sempre ao meu lado.

Ao CESAR que me acolheu desde o inicio do curso e me proporcionou experiências super construtivas para minha carreira profissional.

Por fim agradeço a todos da Universidade Federal Rural de Pernambuco, todos os professores e amigos que fiz nesses 4 anos de curso.

Resumo

O mercado de jogos digitais encontra-se em expansão, onde cada vez mais empresas buscam métodos para expandir seus jogos de maneira mais eficiente e automática, alocando a menor quantidade de pessoas possível. Geração procedural de conteúdos é um tema de constante pesquisa por essas empresas pois ela permite a criação de grande quantidade de conteúdo a partir de arquivos de tamanhos mínimos. A geração de conteúdos proceduralmente pode ser utilizada na criação de cenários, mapas, texturas, roteiros, personagens, entre outros, com pouca ou nenhuma participação humana.

Outra técnica utilizada bastante atual é a Realidade Aumentada, enquanto em Realidade Virtual o usuário visualiza um ambiente totalmente virtual, em Realidade Aumentada o usuário visualiza um ambiente real aumentado com elementos virtuais. Esse é outro fator que está em voga pois recentemente tivemos a popularização de dispositivos e recursos para acessar tal tecnologia, como o Microsoft Hololens ou o Google Glass. Além disso, temos grandes empresas entregando kits de desenvolvimento para Realidade Aumentada em seus sistemas de fábrica, como o Apple ARKit e o Google ARCore, assim permitindo que desenvolvedores criem recursos de Realidade Aumentada e os distribuam de forma fácil e convencional nas lojas App Store e Google Play.

Esse trabalho realiza um estudo sobre as tecnologias de geração de conteúdo procedural e Realidade Aumentada aplicadas a jogos digitais a fim de aprimorar a experiência do usuário criando jogos proceduralmente e exibindo eles em um cenário real em que o jogador está inserido. Um estudo de caso foi desenvolvido utilizando essas duas tecnologias. O estudo apresenta a geração procedural de um jogo de defesa de torres a partir do rastreamento do ambiente em que o jogador está inserido via marcadores de Realidade Aumentada, permitindo o usuário adicionar marcadores no seu ambiente que são interpretados pelo jogo como torres de defesa no jogo, as torres adicionadas devem impedir os inimigos de alcançarem o objetivo.

Palavras-chave: realidade aumentada. geração procedural de conteúdos. jogos digitais

Abstract

The digital gaming market is expanding, where more and more companies are looking for ways to expand their games more efficiently and automatically, by allocating as few people as possible. Content procedural generation is a subject of constant research by these companies because it allows the creation of large amount of content from files of minimum sizes. The generation of content can be used procedurally in the creation of scenarios, maps, textures, scripts, characters, among others, with little or no human participation.

Another technique used quite current is Augmented Reality, while in VR the user visualizes a totally virtual environment, in Augmented Reality the user visualizes a real environment enhanced with virtual elements. This is another factor that is in vogue since we recently had the popularization of devices and resources to access such technology, such as Microsoft Hololens or Google Glass. In addition, we have large companies delivering Augmented Reality development kits to their factory systems, such as Apple ARKit and Google ARCore, thereby allowing developers to create Augmented Reality features and distribute them easily and conventionally in App Store and Google stores Play.

This work performs a study of procedural content generation and Augmented Reality technologies applied to digital games in order to improve the user experience by creating games procedurally and viewing them in a real scenario in which the player is inserted. A case study was developed using these two technologies. The study presents the procedural generation of a tower defense game from the tracking of the environment in which the player is inserted via Augmented Reality markers, allowing the user to add markers in their environment that are interpreted by the game as towers of defense in the game, the added towers must prevent the enemies from reaching the goal.

Keywords: augmented reality. procedural content generation. digital games

Lista de Ilustrações

Lista de Algoritmos

Sumário

1 Introdução

O mercado de jogos digitais se encontra em constante crescimento e os números da indústria são cada vez maiores (NEWZOO, 2017). Realidade Aumentada (RA) e Realidade Virtual (RV) em jogos são temas que também estão em voga nos dias atuais (ANALYSTS, 2017), visto a comercialização e popularidade de dispositivos que reproduzem esse tipo de conteúdo, como o Oculus Rift (OCULUS, 2017) e o Microsoft HoloLens (MICROSOFT, 2017a).

RV coloca o usuário totalmente dentro de um ambiente virtual, enquanto a RA leva conteúdos digitais para o mundo real onde o usuário está inserido (AZUMA, 1997) através de algum dispositivo com câmera, como um celular, computador com webcam, ou mais recentemente com dispositivos específicos para isso como o Microsoft HoloLens e Google Glass (GOOGLE, 2017).

Com a popularidade cada vez maior dos jogos digitais, surgiu-se a necessidade de criar jogos com cada vez com mais tempo de jogabilidade, com uma experiência personalizada para manter o jogador focado no jogo o maior tempo possível. Uma técnica que pode ajudar a resolver esse problema é a geração de conteúdo procedural (procedural content generation - PCG) (TOGELIUS et al., 2011), que permite a criação de conteúdo automático, como níveis, missões, história, recompensas, etc.

Algumas técnicas foram criadas para utilizar o ambiente real do jogador como plataforma de jogos em RA. Por exemplo, o trabalho descrito em (AZAD et al., 2016) faz uma varredura do ambiente real com algum sensor específico, como o Kinect (MICROSOFT, 2017b), para reconstruir o cenário em 3D, permitindo usar os móveis do cenário como nós num grafo, onde pode-se criar plataformas entre os nós caso a distância entre eles seja maior que um limiar definido.

Técnicas mais comuns também são utilizadas, como a utilização de marcadores, que podem ser espalhados pelo ambiente e ser reconhecidos pelos jogos para serem utilizados no desenho de objetos do jogo na posição definida pelos marcadores. Assim sendo, visando criar experiências cada vez mais imersivas para manter o jogador focado por mais tempo, esse trabalho propõe o uso de técnicas para rastreamento de objetos em cena e um estudo de novas interações para criar jogos cada vez mais interessante para os jogadores.

1.1 Problema de Pesquisa

Criar experiências mais imersivas para os jogadores é um estudo constante na área de jogos digitais. RV é uma tecnologia que está sendo bastante pesquisada pela

possibilidade de levar o jogador para dentro do mundo virtual.

O trabalho descrito em (AZAD et al., 2016) é um bom exemplo de como utilizar RA de forma mais interessante para os jogadores, visto que ele leva os jogos digitais para dentro do ambiente que o usuário está inserido. Para realizar essa imersão alguns passos são necessários.

O primeiro passo é o reconhecimento do ambiente onde o jogador está inserido. Utilizando RA em dispositivos móveis ele pode estar inserido em qualquer ambiente da sua casa, trabalho, escola, etc. Existem diversas maneiras de realizar o reconhecimento do ambiente onde o jogador está inserido. O mais comum e simples é a utilização de marcadores. A ideia da utilização de marcadores é espalhar imagens que são facilmente reconhecidas pelas câmeras através de algum arcabouço específico para esse processo, como o Vuforia (VUFORIA, 2017) e ARToolKit (ARTOOLKIT, 2017). Esses marcadores geralmente são imagens em preto e branco que contém um padrão facilmente reconhecido, como um código de barras ou código QR. O Vuforia também trabalha com marcadores mais complexos, como marcadores de imagens e objetos 3D (Dos Santos; DOURADO; BEZERRA, 2016).

Outro método para reconhecimento de ambientes é a utilização de técnicas de processamento de imagem e visão computacional, onde através da imagem da câmera do dispositivo são aplicadas diversas técnicas para reconhecimento de linhas, manipulação de cores ou detecção de formas a fim de descobrir uma área que possa ser jogável. Os trabalhos detalhados em (GEIGER; ZIEGLER; STILLER, 2011) (ENGEL; STURM; CREMERS, 2013) utilizam sequências de imagem para fazer uma reconstrução de um ambiente, criando uma nuvem de pontos que se aproxima dos objetos detectados, criando um escaneamento 3D dos objetos que estão na cena.

A técnica utilizada em (AZAD et al., 2016) é similar à técnica mostrada anteriormente, porém ela utiliza o Kinect para criar uma aproximação mais fiel do ambiente. Utilizando essa ferramenta os resultados são melhores, pois além da utilização da câmera o Kinect tem sensores de profundidade que ajudam no reconhecimento do ambiente, como mostrado mais detalhadamente no trabalho em (MOHAMMADKHANI, 2013).

Ao utilizar algumas dessas três técnicas para o reconhecimento do ambiente o próximo passo é a geração do conteúdo baseado no ambiente reconhecido. Um grafo pode ser gerado contendo os objetos encontrados no reconhecimento e que serão utilizados como possíveis plataformas no jogo. O trabalho detalhado em (AZAD et al., 2016) utiliza a distância dos nós, pesos das arestas do grafo, para verificar se o jogador consegue pular de um móvel para o outro ou se uma plataforma é necessária. Técnicas de pathfinding como A* (CUI; SHI, 2015) podem ser utilizadas para verificar qual é o melhor caminho entre os nós. O trabalho descrito em (AZAD et al., 2016) vai além ao também considerar que o movimento do jogador é um fator na geração do conteúdo dentro do ambiente,

considerando que o usuário pode se locomover para se aproximar de móveis que estão distantes.

Esse trabalho tem como problema de pesquisa a criação de métodos para auxiliar a geração de jogos em ambientes reais com RA, como reconhecer o ambiente e projetar jogos personalizados para cada jogador, como fazer o jogador interagir com objetos reais e como fazer jogos que usam RA para expandir o cenário real do jogador e trazer novas interações para ele.

1.2 Objetivos

Tendo em vista o problema de pesquisa citado na seção anterior, os objetivos deste trabalho são detalhados a seguir.

1.2.1 Objetivo Geral

O presente trabalho possui como objetivo geral criar funções que auxiliem na geração de conteúdos procedurais baseados no cenário real em que o usuário está inserido, com aplicações em RA e jogos digitais.

1.2.2 Objetivos Específicos

Os seguintes objetivos específicos podem ser citados:

- Reconhecer o ambiente físico do jogador.
- Gerar conteúdo procedurais baseado no cenário físico do jogador.
- Criar interações do jogador com conteúdo em RA.

1.3 Metodologia

Aqui é descrito o processo de desenvolvimento do presente trabalho, o qual iniciou-se com a proposta do orientador Prof. João Paulo Lima de realizar um estudo de caso baseado no trabalho descrito em (AZAD et al., 2016), no qual utiliza-se RA e conteúdos gerados proceduralmente em dois jogos de exemplo. A partir dessa referência as linhas de pesquisas foram definidas: RA e PCG.

Um estudo bibliográfico foi realizado acerca das linhas de pesquisa definidas acima, sendo investigados trabalhos como (COOK, 2015), (TUTENEL; SMELIK, 2009). Um estudo dos algoritmos foi realizado baseado nas pesquisas. A maioria dos artigos utiliza computação evolucionária para criar os conteúdos proceduralmente.

O Vuforia foi escolhido como pacote de desenvolvimento de software (SDK) de RA, utilizando o Unity. A integração do Vuforia com o Unity é interessante na tarefa de desenvolver jogos 3D utilizando RA.

Um protótipo de defesa de torres foi desenvolvido utilizando o Unity 3D e o Vuforia. Diferentemente de (AZAD et al., 2016), escolhemos utilizar marcadores de borda e de imagem no rastreamento do ambiente no lugar do rastreamento por escaneamento com o Kinect pois tal método é mais custoso computacionalmente e depende do usuário ter um Kinect e de escanear todo ambiente que está inserido. Os marcadores de borda e de imagem são simples de conseguir e os kits de desenvolvimento já estão preparados para dar suporte a este tipo, agilizando o desenvolvimento e a utilização por parte do usuário final.

1.4 Estrutura do Trabalho

Os seguintes capítulos estruturam-se inicialmente com o levantamento dos trabalhos relacionados como forma de justificativa e posteriormente com a abordagem proposta. O Capítulo 3 abordará os conceitos básicos e as terminologias utilizadas no trabalho, definição do que é RA, alguns diversos tipos de marcadores e PCG. O Capítulo 4 mostra a solução desenvolvida, detalhando todas as ferramentas como o Unity 3D, Vuforia e os algoritmos como malha de navegação (navigation mesh - NavMesh) e o algoritmo de geração de árvore de extensão mínima (minimum spanning tree - MST) e no final desse capítulo são apresentados os resultados encontrados nesse estudo de caso. O trabalho termina no Capítulo 4 com a conclusão e propostas para trabalhos futuros.

2 Trabalhos Relacionados

Trabalhos relacionados na área de PCG tem por objetivos descobrir novas técnicas para aprimorar os resultados de pesquisas anteriores ou de usar os algoritmos existentes em um novo contexto, desde a geração de mapas para jogos de tiro em primeira pessoa como a geração de todos os elementos para criar um jogo totalmente procedural.

O trabalho descrito em (COOK, 2015) usa a técnica de PCG para criar ambientes em 3D usando algoritmos de computação evolucionária considerando restrições de visibilidade. É um estudo de como realizar level design para jogos sem que eles sejam totalmente aleatórios. Ele cria ambientes considerando restrições do tipo: dois jogadores devem enxergar um ao outro, ou que o jogador apenas pode visualizar a saída, mas não seus inimigos.

TOGELIUS et al. (2011) faz uma grande avaliação da área de PCG, usando algoritmos evolucionários não só na criação de mapas procedurais, mas de qualquer tipo de conteúdo em jogos, como personagens não jogáveis (non-playable characters - NPCs), terrenos, mapas, níveis, estórias, diálogos, missões, personagens, regras, dinâmicas e armas.

Conceitos de semântica para organizar e gerar cenas em ambientes virtuais é utilizado em (TUTENEL; SMELIK, 2009). Ele utiliza o conceito de ontologias e de descrição de recursos (resource description framework - RDF) para criar uma estrutura hierárquica entre os elementos que estarão na cena gerada, como por exemplo: numa sala de estar temos que posicionar o sofá em frente a televisão e deve-se existir um espaço para circulação entre eles. O projeto gera uma planta baixa do ambiente com as restrições de espaço e circulação, como o espaço para abrir a porta ou o espaço para o uso do sofá. A partir dessa planta baixa ele gera um ambiente 3D, escolhendo os modelos 3D numa biblioteca de arquivos, podendo variar entre os conteúdos selecionados. O uso dessa biblioteca no projeto disponibiliza uma grande quantidade de modelos, conseguindo gerar cenas que se diferenciam umas das outras e que tenham temas, como por exemplo gerar um ambiente com decorações de natal.

O trabalho detalhado em (GOESELE; STUERZLINGER, 1999) tem uma proposta similar ao de (TUTENEL; SMELIK, 2009). Ele utiliza restrições de semântica para posicionar objetos em cena, juntamente com restrições de área de superfície, ou seja, ele considera quais áreas dos objetos podem receber objetos sobre e qual área desse objeto deve tocar o chão, parede ou teto. Então ele cria um grafo que representa uma cena virtual apenas informando, por exemplo, que um dado móvel deve ficar junto a parede e em cima dele deve ter um vaso ou uma televisão.

Algoritmos evolucionários para criar mapas de jogos de tiro em primeira pessoa

é utilizado em (CARDAMONE et al., 2011). Ele gera 4 tipos diferentes de mapa com diferentes representações de cromossomo e faz avaliações sobre qual deles tem a melhor avaliação dentro de um cenário real com bots jogando nos mapas criados. A maioria dos trabalhos de PCG utiliza jogos de plataformas e esse traz uma abordagem mais complexa, mostrando o poder dos algoritmos evolucionários. A Figura 1 mostra o resultado obtido na geração de um mapa procedural para jogos de tiro em primeira pessoa.

Figura 1 – Mapa de jogo de tiro em primeira pessoa. Fonte: (CARDAMONE et al., 2011)

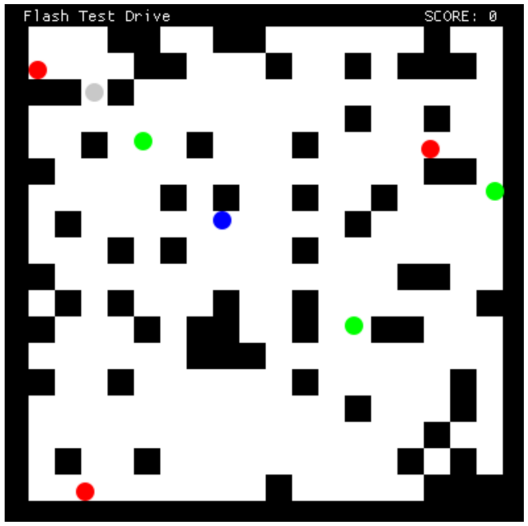

Figura 2 – Jogo gerado proceduralmente. Fonte: (COOK; COLTON, 2011)

O trabalho detalhado em (COOK; COLTON, 2011) tem uma proposta bastante ousada que é a geração procedural de um jogo por completo, não apenas mapas ou plataformas. Ele gera o mapa, regras, posições dos personagens e NPCs. Através de

múltiplos algoritmos evolutivos ele consegue evoluir cada item necessário para o jogo e juntar eles coerentemente para ter um jogo jogável. O projeto foi desenvolvido para gerar jogos em 2D extremamente simples, porém hoje já consegue desenvolver jogos com uma interface mais evoluída. A Figura 2 mostra um dos jogos gerados pelo algoritmo desenvolvido no trabalho, o jogador é o ponto cinza e os NPCs são os pontos vermelhos, azul e verdes.

O artigo apresentado em (HORSWILL; FOGED, 2012) utiliza de restrições para garantir que um jogo que utiliza PCG seja balanceado, como garantir que a quantidade de vida em um cenário é suficiente para o jogador conseguir finalizar ele. Ele utiliza jogos do estilo roguelike nos seus testes por ser jogos que tem um uso de conteúdo procedural elevado, pois cada sala do jogo pode ser totalmente aleatória, contendo inimigos, vidas, chaves, portas, entre outros itens totalmente gerados por algoritmos de PCG.

(COMPTON; MATEAS, 2006) propõe um método para criar conteúdo procedurais para jogos de plataforma, visto que esse é um tipo de jogo que não conseguiu explorar as vantagens do conteúdo procedural, diferente de jogos de estratégia ou RPG onde temos jogos comercialmente famosos que usam a técnica como Civilization (2K, 2017) e Diablo II (BLIZZARD, 2017). O autor fala do ritmo que os jogos de plataforma têm que dificultam a utilização da técnica, visto que qualquer alteração no jogo afeta a forma que a física é aplicada, a velocidade que o jogador está correndo ou a altura do pulo necessária para completar alguma ação. Para resolver esse problema ele utiliza gramáticas para representar os níveis, onde pode-se quebrar em várias células a fim de gerar um mapa que é a composição de vários elementos, como plataformas, canos, inimigos, etc. Ele faz várias verificações das possíveis trajetórias que o jogador vai realizar para pular de uma plataforma para outra para verificar se a disposição delas é viável para o jogador.

O artigo apresentado em (AZAD et al., 2016) utiliza PCG para jogos em RA para criar experiências personalizadas para os jogadores de acordo com o ambiente que eles estão inseridos. Ele faz o reconhecimento do ambiente com o Kinect, gerando um modelo 3D do mesmo. Após reconhecer o ambiente ele transforma os móveis disjuntos em nós em um grafo. Os nós mais distantes são utilizados para criar plataformas para o jogador se locomover entre os móveis. Dois protótipos de jogos foram criados, um deles baseado no jogo Lemmings (WIKIPéDIA, 2018), onde os personagens se movem sobre um caminho e ao colidir com botões virtuais eles mudam de direção. O jogador deve posicionar os botões para os personagens irem até o objetivo final. O jogo original era de plataforma, os personagens se moviam automaticamente e o jogador devia interagir com o jogo para os personagens realizarem algumas ações como pular, saltar, entre outros. O protótipo gerado utilizou as mecânicas do jogo original transferidas para 3D e com cenário baseado no mundo real do jogador. O outro protótipo criado é baseado no famoso jogo Super Mario Bros (NINTENDO, 2017) onde o jogador controla um avatar virtual e deve saltar sobre

os móveis da sala para conseguir ir até o objetivo final. Plataformas virtuais são geradas proceduralmente no caminho do jogador para que ele consiga alcançar o objetivo e vários inimigos são colocados para que o jogador se sinta mais desafiado a completar as ações, utilizando a mecânica de movimentação dentro do ambiente. O trabalho utiliza técnicas de pathfinding para achar o melhor caminho para criar o PCG, tanto na movimentação virtual como no mundo real. Visto que câmera deve se mover para alcançar móveis mais distantes, o algoritmo leva em consideração que o jogador deve se movimentar dentro do ambiente, o que gera uma complexidade a mais pois o rastreamento do ambiente deve levar em consideração que a câmera de RA estará em movimento. A Figura 3 mostra duas possibilidades de plataformas geradas para o protótipo do Mario, os objetos em cena como mesa, poltrona e sofá foram gerados com o sensor do Kinect, criando uma representação 3D do ambiente que o jogador está inserido.

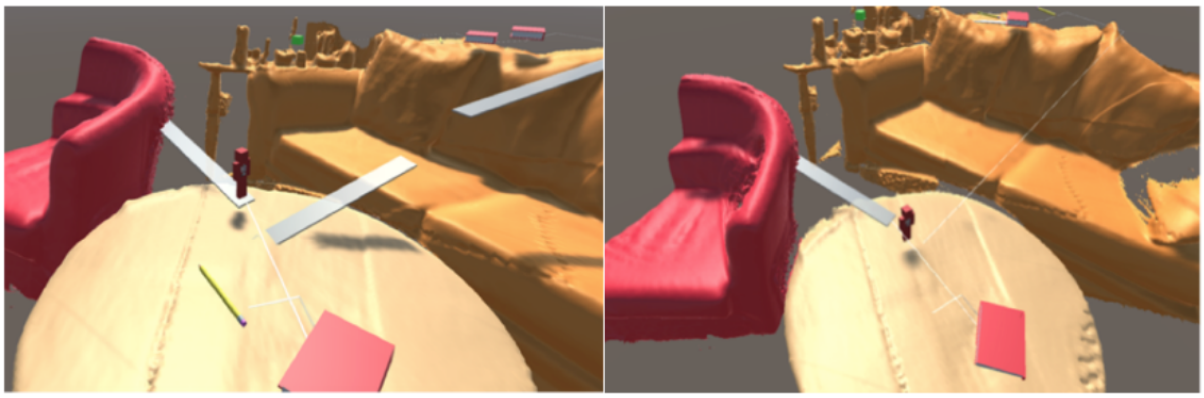

Figura 3 – Plataformas virtuais. Fonte: (AZAD et al., 2016)

Poucas pesquisas foram realizadas na área de detecção de ambientes reais na utilização em jogos digitais com RA, como por exemplo o trabalho descrito em (AZAD et al., 2016) que utiliza RA para gerar dois protótipos de jogos que usam o ambiente real do jogador como cenário. Ele utiliza o Kinect para fazer um reconhecimento do ambiente e gerar um modelo 3D que é usado para gerar as plataformas virtuais que serão usadas pelo jogador, além de criar algumas interações com objetos virtuais. Esse trabalho pretende expandir os conceitos visto pelo artigo de AZAD et al. (2016), fazendo um reconhecimento do ambiente para verificar se existe algum objeto que pode ser usado para melhorar o conteúdo a ser gerado e também algoritmos de inteligência artificial para gerar inimigos que fiquem mais inteligentes para prender a atenção do jogador por mais tempo.

3 Conceitos Básicos e Terminologia

Nesta capítulo veremos alguns conceitos básicos encontrados neste trabalho tais como: o que é RA, quais os marcadores que são utilizados nessa tecnologia e como técnicas de PCG são utilizadas em jogos digitais.

3.1 Realidade Aumentada

RA é utilizada em uma grande quantidade de aplicações hoje em dia e existem diversas maneiras de aplicar suas técnicas em vários cenários diferentes, como aplicações para rastreamento de faces, jogos digitais, visualização de conteúdo 3D como plantas baixas, entre outros.

3.1.1 Definição

RA é a inserção de conteúdos digitais em um cenário real. Se diferencia da RV, que insere o usuário em um mundo totalmente digital onde o usuário não consegue ver a realidade que está em sua volta. RA permite ao usuário ver o mundo real em sua volta e aumentar ele, inserindo conteúdos virtuais visíveis apenas por algum dispositivo específico. Na Figura 4 nós podemos ver um exemplo de RA, onde o usuário está vendo em cima da mesa um personagem digital. Nesse caso ele está usando o Microsoft Hololens, que é um Head-Mounted Display (HMD). Apesar de apenas um display com uma câmera de vídeo seja suficiente para visualizar o conteúdo de RA, os HMD são óculos que fornecem uma imersão maior, onde o usuário consegue ver os objetos digitais diretamente nos óculos.

Figura 4 – RA com Hololens. Fonte: (APTIV8, 2018)

Na Figura 5 é possível ver que não existe uma oposição entre RA e RV, elas se complementam no contínuo chamado de Realidade-Virtualidade. Da esquerda para a direita nós temos o mundo real que conhecemos hoje, seguido do que conhecemos por RA, que é a inserção de conteúdos digitais em um mundo real com predominância de elementos reais e em tempo real. Continuando mais à direita temos a Virtualidade Aumentada que seria a inserção de conteúdo reais em um mundo virtual com predominância de elementos virtuais também em tempo real. Finalmente, temos com um mundo totalmente virtual que chamamos de RV. RA não precisa necessariamente de dispositivos como HMD. (AZUMA, 1997) define que RA possui três características:

1. Combinar real e virtual
2. Interativo em Tempo Real
3. Registrado em 3D

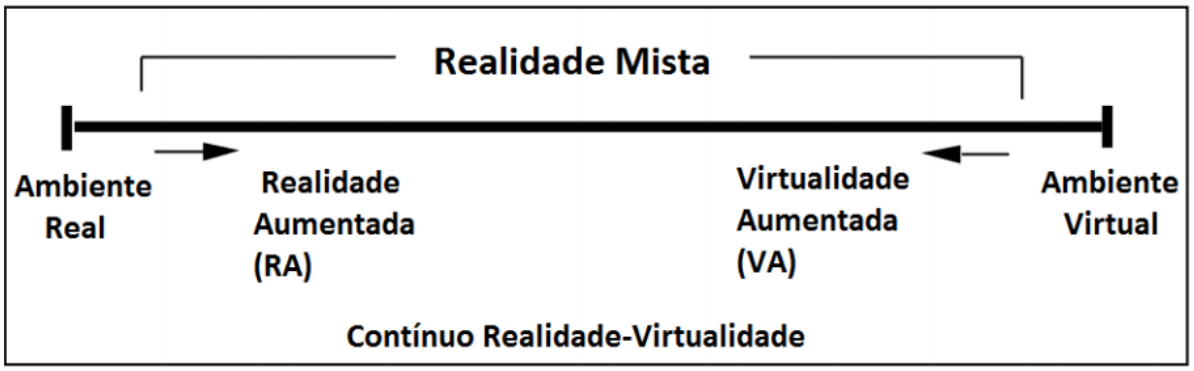

Figura 5 – Relação entre RA e RV no Contínuo Realidade-Virtualidade. Fonte: (MILGRAM HARUO TAKEMURA, 1995)

3.1.2 Rastreamento

Para a RA funcionar com perfeição existe a necessidade de rastrear o ambiente onde o usuário está inserido para identificar quais os locais em que os conteúdos digitais devem ser colocados. Existem diversas maneiras de rastrear um ambiente, sendo uma das mais comuns a utilização de marcadores. Entretanto, também é possível realizar o rastreamento usando características naturais da cena ou com sensores específicos.

3.1.2.1 Rastreamento com Marcadores

Essa é a maneira mais simples de rastrear ambientes para RA, sendo a abordagem mais bem estabelecida na academia, que consiste na definição de uma imagem padrão

que será usada no mundo real. Sempre que a câmera de vídeo apontar para tal marcador a aplicação conseguirá identificá-lo de maneira fácil e desenhar o objeto virtual em uma posição relativa ao marcador.

Figura 6 – Código QR. Fonte: (CONTRIBUTORS, 2018)

Um tipo de marcador que pode ser utilizado em RA é o Código QR (QRCode), Figura 6. pois ele é simples de imprimir e tem uma quantidade de detalhes bem alta, fator importante no reconhecimento. Porém, qualquer imagem pode ser utilizada como marcador, uma vez que ela seja rica em detalhes, tenha um bom contraste e sem padrões repetitivos. Alguns arcabouços como o Vuforia permitem adicionar objetos complexos como marcadores, como cilindros e caixas. No cilindro você deve informar uma imagem referente à lateral do objeto e na caixa você deve informar 6 imagens que compõem as faces do objeto. Na Figura 7. podemos ver uma aplicação do marcador de Código QR, onde após a câmera de vídeo reconhecer a posição do marcador, um objeto 3D é plotado na mesma posição e o rastreamento é iniciado, ou seja, ao mover o marcador no mundo real o objeto plotado também se moverá acompanhando o marcador.

3.1.2.2 Rastreamento sem Marcadores

Apesar de marcadores serem a maneira mais fácil de realizar o rastreamento, eles não são obrigatórios. Diversos aplicativos disponíveis no mercado hoje como o Instagram e Snapchat contam com recursos para sobrepor animações 3D na face rastreada do usuário. Isso é possível com técnicas de processamento de imagem e visão computacional. Esse tipo de rastreamento é dividido em duas tarefas:

1. Detecção da face

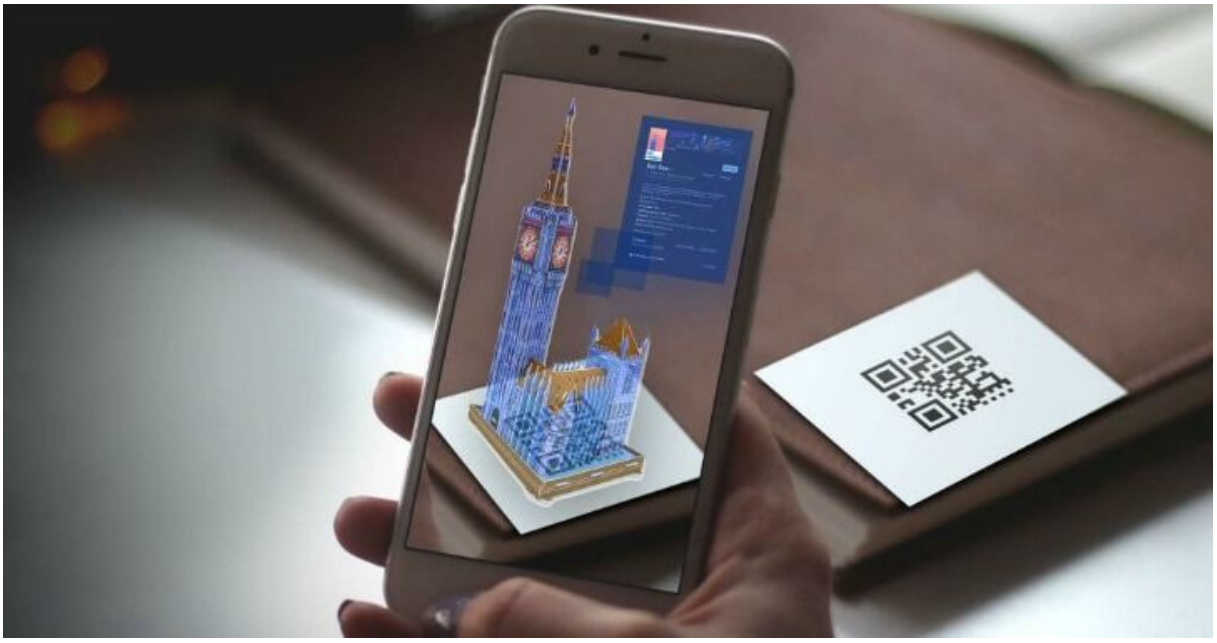

Figura 7 – Aplicação Código QR. Fonte: (SHINDE, 2017)

2. Rastreamento das características da face

Uma terceira tarefa pode ser adicionada caso se deseje reconhecer quem é aquela pessoa que tem aquelas características, caso se esteja trabalhando em uma aplicação de segurança e seja preciso reconhecer quem está entrando no prédio, por exemplo. O trabalho de (HJELM; HJELMÅS; LOW, 2001) mostra algumas dificuldades da tarefa de detecção da face, como a remoção do fundo da imagem, localizar o rosto em diferentes poses, orientações, iluminações, ambientes, entre outros. Existem diversas técnicas para realizar esse processamento, sendo a maioria delas com algoritmos de inteligência artificial e computação evolucionária, como as redes neurais.

No trabalho de (HASSABALLAH; MURAKAMI; IDO, 2013) temos um exemplo de, após a detecção da face realizada, como é possível detectar as características da face, como olhos, boca e nariz. Isso será essencial para se conseguir colocar animações 3D na face do usuário como nos aplicativos citados acima. Na Figura 8 temos um exemplo dos resultados obtidos no trabalho de (HASSABALLAH; MURAKAMI; IDO, 2013). Após detectar a posição do rosto ele usa a proporção áurea para descobrir a posição dos olhos e nariz.

3.1.2.3 Rastreamento com Sensores

O trabalho de (AZAD et al., 2016) utiliza uma abordagem diferente para realizar o rastreamento do ambiente. Ele faz um escaneamento com o Kinect, criando um cenário 3D do local onde o jogador está inserido. A Figura 9 mostra o resultado obtido com o

escaneamento. Apesar de o resultado ser bem satisfatório, é necessário que o usuário possua um sensor como o Kinect e que ele movimente o sensor ao redor do ambiente para conseguir escaneá-lo.

Figura 8 – Características da face. Fonte: (HASSABALLAH; MURAKAMI; IDO, 2013)

Figura 9 – Rastreamento com Kinect. Fonte: (AZAD et al., 2016)

3.2 Geração Procedural de Conteúdo

Com a popularização dos jogos, cada vez mais existe a necessidade de expandir a experiência do usuário. Segundo a (ESA 2010), em 2009 69% dos americanos jogavam vídeo games ou jogos de computador em suas casas, a média de idade dos jogadores era de

35 anos e a média da experiência com jogos eletrônicos era de 12 anos. Podemos observar que jogos são um fator importante no entretenimento e por isso técnicas como PCG vêm sendo estudadas tanto na academia como no mercado.

Técnicas procedurais estão em voga pela praticidade que elas trazem no desenvolvimento de jogos. Com ela é possível desenvolver jogos complexos com uma equipe reduzida, com uma quantidade limitada de tempo e recursos, e para manter a qualidade desejada ainda podemos ter designers validando e refinando todas os conteúdos gerados proceduralmente.

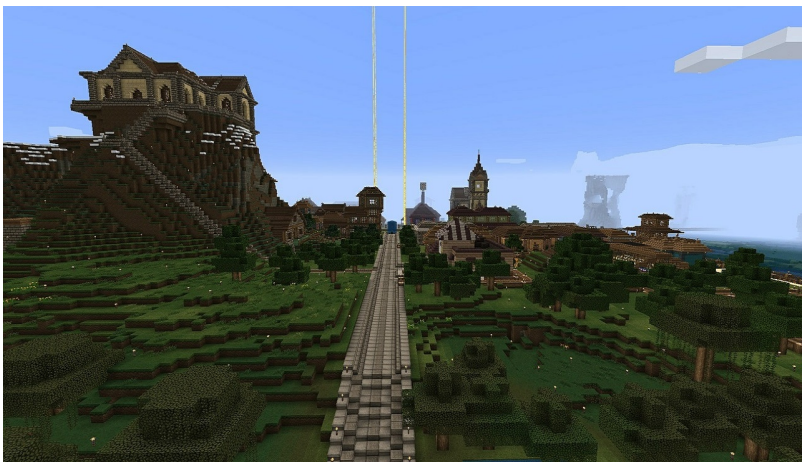

Figura 10 – Minecraft. Fonte: (COELHO, 2015)

Na Figura 10 vemos um exemplo de jogo que utiliza essa técnica, o jogo Minecraft, que consegue a partir de números aleatórios gerar um mundo enorme seguindo várias restrições geográficas e físicas, como encontrar mais árvores em biomas de floresta e bem menos em biomas desérticos. Técnicas procedurais são frequentemente utilizadas na criação de mapas aleatórios como no Minecraft, porém tais técnicas podem ser utilizadas em diversas áreas dos jogos, como a criação de enredos, personagens, texturas, entre outros. A Figura 11 mostra o jogo No Man's Sky, que consegue gerar um sistema completo totalmente aleatório, como planetas, animais, vegetação, entre outros. Segundo os desenvolvedores o jogo tem 18 quintilhões de planetas, todos eles com fauna e flora própria.

Figura 11 – No Man's Sky. Fonte: (GRIXTI, 2017)

4 Solução de Geração Procedural de Conteúdo para Jogos de Realidade Aumentada

O presente capítulo apresenta a descrição detalhada do estudo de caso desenvolvido para esse trabalho. Foram utilizados alguns algoritmos como a MST (EPPSTEIN, 1996) e o NavMesh (CUI; SHI, 2015), que foram desenvolvidos utilizando C# com o Unity (UNITY, 2018) e o Vuforia (VUFORIA, 2017). O Unity foi escolhido pela fácil curva de aprendizado e pelas facilidades de se desenvolver jogos em 3D com uma interface gráfica amigável. O Vuforia tem uma boa integração com o Unity, permitindo a criação de marcadores de diversos tipos diretamente da interface. As próximas seções explicam detalhadamente cada tecnologia utilizada.

4.1 Unity

Também conhecido como Unity 3D, é um software especializado em renderização de cenas majoritariamente 3D e em tempo real e um ambiente de desenvolvimento integrado criado pela Unity Technologies. Ele se popularizou ao dar ênfase na criação de jogos multiplataformas, dando suporte a PC, MacOS, iOS, Android, Xbox, Playstation, entre outras, e possuindo duas linguagens de codificação, o C# e o JavaScript.

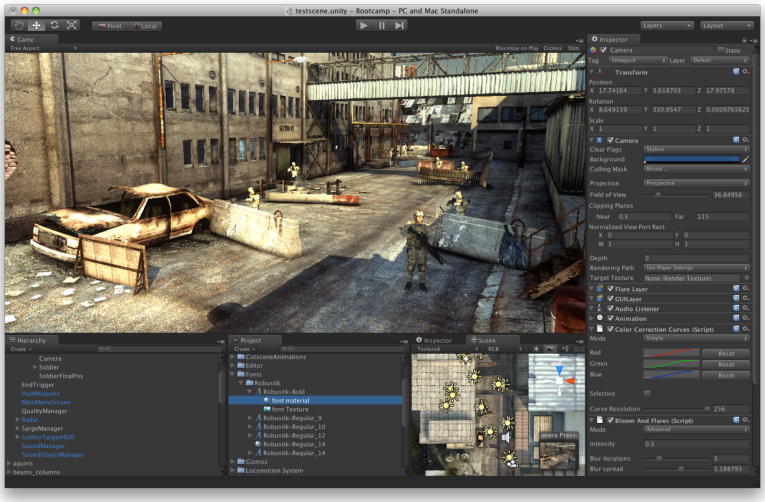

Figura 12 – Unity 3D. Fonte: (UNITY, 2018)

Na Figura 12 temos a interface gráfica do Unity 3D, onde além da visualização do jogo na parte central temos diversos painéis que nos auxiliam no momento do desenvol-

vimento, como a hierarquia de objetos em cena, navegação de arquivos do projeto e um painel de inspeção que permite atualizar propriedades de qualquer objeto selecionado.

4.2 Vuforia

O Vuforia é um kit de desenvolvimento de software para RA. Ele utiliza visão computacional para rastrear diversos tipos de marcadores, como marcador de borda, similar aos códigos QR, alvos de imagem, de multiobjetos, cilindros e objetos 3D. Através do portal de desenvolvedor do Vuforia, é possível criar novos alvos, onde ele avaliará se a imagem que queremos utilizar como alvo é boa o suficiente para conseguirmos um bom rastreamento. Isso é possível porque ele realiza alguns processamentos na imagem e consegue descobrir o contraste dela, se existem padrões repetitivos e uma boa quantidade de detalhes. A saída dessa análise será como a Figura 13: quanto mais pontos amarelos na imagem melhor, pois são os pontos que o Vuforia utilizará para rastrear o alvo em questão.

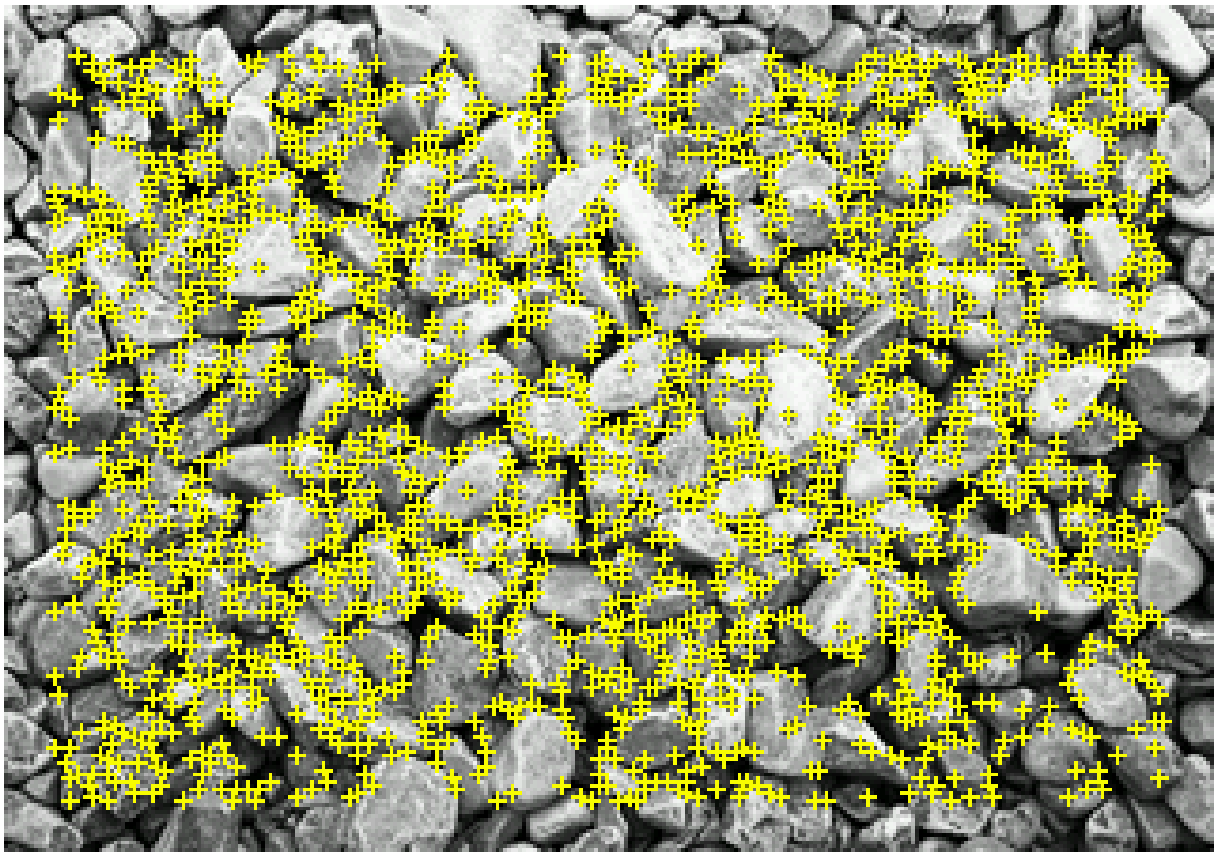

Figura 13 – Alvo de imagem analisado pelo Vuforia. Fonte: (VUFORIA, 2017)

4.3 Árvore de Extensão Mínima

A MST é um conceito utilizado em grafos para encontrar o menor gráfico que pertence ao gráfico original e que conecta todos os vértices com o menor custo possível. A definição de árvore de extensão diz: dado um grafo não direcionado e conectado G=(V, E), uma árvore de extensão do grafo G é uma árvore que contém todos os vértices de G e é um subgrafo de G, ou seja, todas as arestas da árvore pertencem a G. Grafos podem ter diversas árvores de extensão, o custo dessa árvore é a soma dos pesos de todas as arestas. Uma MST é a árvore de extensão que contém o menor custo dentre todas as possíveis árvores.

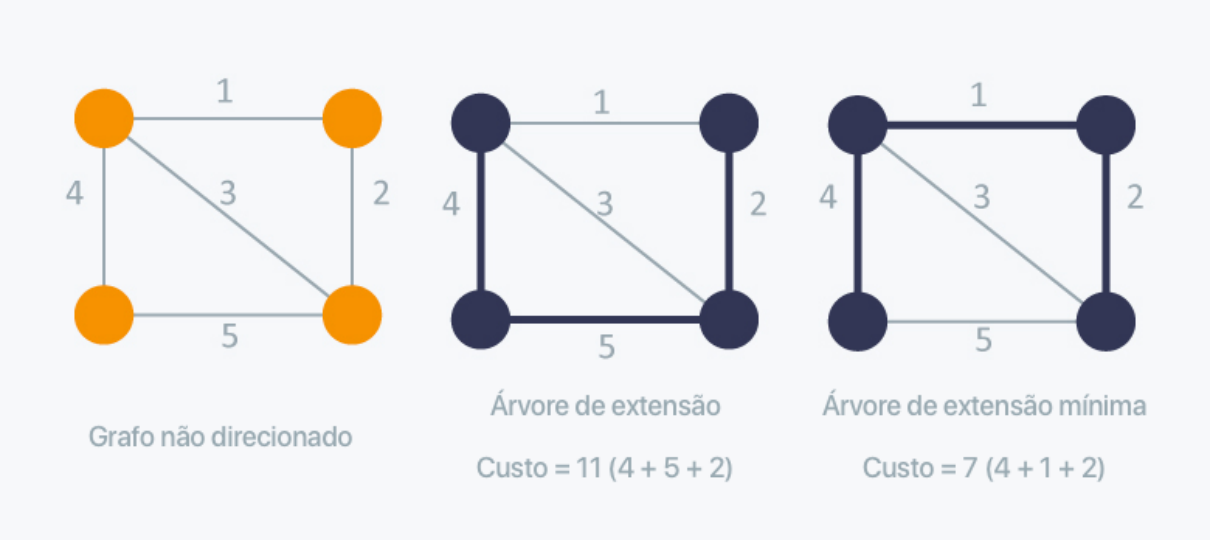

Figura 14 – Grafo e suas árvores de extensão. Fonte: (HACKEREARTH, 2018)

Na Figura 14 temos um exemplo de grafo não direcionado à esquerda e duas árvores de extensão mais à direita. Como podemos ver, o mesmo grafo pode ter mais de uma árvore com custos diferentes, sendo que a menor árvore de extensão que passa por todos os vértices é a mais à direita. Entre os dois famosos algoritmos para resolver o problema da criação da MST, algoritmo de Kruskal e algoritmo de Prim, foi escolhido utilizar o algoritmo de Prim pela facilidade de implementação.

4.3.1 Algoritmo de Prim

O algoritmo de Prim foi escolhido nesse projeto pela sua facilidade de implementação, no qual $\pi[v]$ indica o predecessor de v. Após o término do algoritmo, para cada v pertencente aos vértices de G, $\pi[v] \rightarrow v$ representa uma aresta selecionada para a árvore geradora mínima se $\pi[v] \neq nulo$. O algoritmo retorna o conjunto dessas arestas, formado pelos pares

$(\pi[v], v)$. Q é um conjunto de pares (peso, vértice). O método extrairMin(Q) deve extrair o menor elemento de Q; um par (a,b) é menor que um par (c,d) se a $<$ c ou se a $=$ c e b $<$ d. S é um conjunto que armazena os vértices cujas adjacências já foram analisadas.

Algorithm 1 – Algoritmo de Prim

```
1:  procedure PRIM(G)                                          ▷ G é o grafo
2:      s ← selecionaUmElemento(vertices(G))                   ▷ Escolhe o vértice inicial
3:
4:      for v ϵ vertices(G) do                                 ▷ Para cada vértice em G
5:          π[v] ← nulo
6:
7:      Q ← {(0, s)}
8:      S ← ∅
9:
10:     while Q ≠ ∅ do
11:         v ← extraiMin(Q)
12:         S ← S ∪ {v}
13:
14:         for u adjacente a v do
15:             if u ∉ S e pesoDaAresta(π[u] → u) > pesoDaAresta(v → u) then
16:                 Q ← Q\{(pesoDaAresta(π[u] → u), u)}
17:                 Q ← Q ∪ {(pesoDaAresta(v → u), u)}
18:                 π[v] ← v
19:
20:     retorna {(π[v], v) | v ϵ vertices(G) e π[v] ≠ nulo}
```

4.4 Malha de Navegação

Buscar caminhos é uma grande área da inteligência artificial usada pelos jogos hoje em dia. A maioria dos jogos utilizam algum algoritmo para calcular a melhor rota entre dois pontos. A* é um algoritmo famoso tanto na industria como na academia pela sua facilidade de implementação e baixo custo de memória, porém algoritmos mais recentes podem ajudar a melhorar os resultados obtidos como por exemplo o NavMesh.

NavMesh é uma técnica para representar o mundo em forma de polígonos ou triângulos e a partir deles procurar o melhor caminho com qualquer algoritmo de busca em grafo, como a A* por exemplo, visto que dentro dos polígonos existe uma área livre de obstáculos. Essa malha de polígonos pode ser criada automaticamente ou manualmente pelo *Level Designer* na edição do mapa. Os polígonos devem ser convexos, ou seja, todo segmento de reta que conecta dois pontos dentro do polígono passa por dentro do polígono.

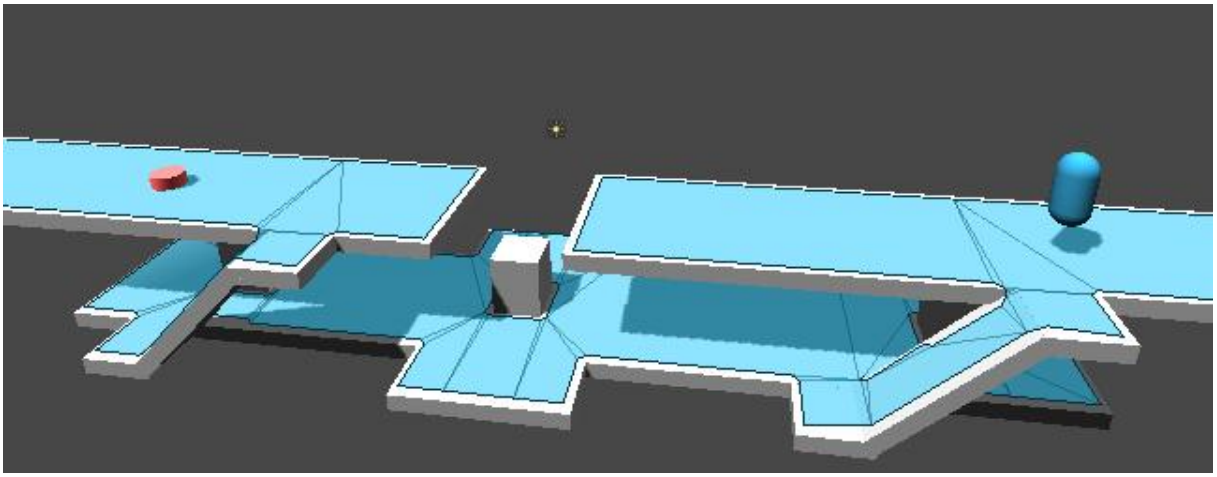

Figura 15 – NavMesh do Unity 3D. Fonte: (TSUBAKI, 2012)

Na Figura 15 vemos um NavMesh criado pelo Unity 3D, onde a área em azul é a área navegável pelos agentes, logo eles estão impedidos de andarem nas bordas das plataformas e de atravessar o obstaculo no centro inferior, por exemplo.

4.5 Implementação

Após detalhar todas as ferramentas e tecnologias utilizadas nesse trabalho, esta seção mostra o protótipo desenvolvido para o estudo do caso. Após diversos testes foi desenvolvido um jogo de defesa de torres, onde o jogador deve eliminar os inimigos antes de eles chegarem no final do percurso. Apesar de ser um jogo simples, ele tira proveito de todos os conceitos utilizados nesse trabalho, como o conteúdo gerado proceduralmente na criação de plataformas e da RA para integrar o jogo ao ambiente do usuário.

O objetivo do jogo de defesa de torres é impedir que os inimigos cheguem ao objetivo por meio de armadilhas e torres que são dispostas no mapa para atrasar-los e que atiram neles enquanto passam. Normalmente os inimigos percorrem um caminho fixo no mapa, saindo de um ponto de origem em ondas de inimigos, um grupo de inimigos que ficam mais difíceis com o avanço do jogo, e as torres só podem ser colocadas em regiões pré-determinadas, como o jogo Kingdom Rush (IRONHIDE, 2011), porém diversas variações podem ser encontradas nos jogos hoje em dia, o jogo Fieldrunners (SUBATOMIC, 2008) por exemplo, permite os jogadores colocarem torres em qualquer lugar no campo de jogo, criando um labirinto que será percorrido pelos inimigos, sempre procurando o menor caminho. O protótipo desenvolvido segue uma das possíveis variações com o mesmo objetivo, onde os inimigos caminham livres pelas superfícies disponíveis e as torres podem ser dispostas em qualquer área ao redor de cada superfície, os inimigos são gerados inicialmente a cada 8 segundos e a cada 5 inimigos gerados esse tempo diminui meio segundo, tornando o jogo mais difícil. Cada inimigo carrega uma quantidade de moedas

que é recuperado pelo jogador quando eles são destruídos, essas moedas são utilizadas nas compras das torres.

Foi definida a utilização de marcadores de alguns tipos para fazer o rastreamento do ambiente em que o jogador está inserido. Tal escolha foi feita pela não disponibilidade de um sensor como o Kinect para gerar uma representação do ambiente e pela dificuldade de rastrear o ambiente sem a utilização de marcadores, pois seria necessário utilizar técnicas complexas de visão computacional para identificar todas as superfícies planas, o que não é o foco deste trabalho. Foram utilizados alguns tipos de marcadores de RA nesse projeto, como o de imagem e de borda para identificar as superfícies planas. O jogador é responsável por distribuir eles dentro do seu ambiente considerando que o tamanho dos marcadores no momento da impressão deve ser no mínimo a distância da câmera / 10, para a câmera de vídeo conseguir rastrear eles sem dificuldade. Por exemplo, se o alvo tem 20cm ele deve estar a no máximo 2 metros de distância da câmera. Outro marcador utilizado foi o de multialvos e de cilindro para representar os obstáculos e as torres que o jogador deve inserir no jogo. Tal escolha foi feita pois objetos reais podem ser utilizados como marcador, dando ao usuário uma imersão maior ao jogar.

Cada marcador de imagem ou borda rastreado pelo Vuforia é considerado uma superfície no jogo, representando superfície plana do ambiente do jogador, como uma mesa, sofá, cadeira, entre outros. Um grafo é criado após o rastreamento e cada superfície vira um nó em um grafo que será utilizado pelo algoritmo de Prim que foi implementado em C#. A saída desse algoritmo informa quais as pontes devem ser criadas e quais caminhos podem ser ultrapassados pelos inimigos sem a necessidade da plataforma, pulando essas distâncias. Baseado nessa saída primitivas de cubos são geradas e dispostas no jogo em tempo de execução, para dispor essas plataformas no jogo é necessário definir suas posições e calcular as escalas para as plataformas serem dispostas corretamente. A posição de cada plataforma é o ponto médio entre duas superfícies e sua escala é a distância entre elas. Essas plataformas são os conteúdos procedurais, são criados em tempo de execução e se adaptam a qualquer configuração de superfícies definidas pelo usuário. As plataformas criadas são o caminho que os inimigos utilizam para ir do ponto de origem para o ponto de chegada. Esses pontos também são definidos em tempo de execução, sendo que as superfícies mais extremas definidas pelo usuário são escolhidas como ponto de origem e de destino para os inimigos.

Como maneira de interação o jogador pode colocar objetos reais que são identificados como marcadores multialvos ou de cilindro pelo Vuforia e bloquear alguma das pontes criadas pelo algoritmo de Prim, obrigando os inimigos a percorrerem um percurso maior e ficando mais tempo sendo alvo das torres criadas pelo jogador. Existem marcadores para representar as torres e eles podem ser colocados em qualquer lugar visível pela câmera de vídeo. Se o jogador tiver dinheiro suficiente, uma é criada sobre o marcador, atirando em

todos os inimigos visíveis. O jogador acumula moedas que são usadas na compra de torres ao eliminar inimigos com as torres, o jogo é iniciado com uma quantidade suficiente para comprar a torre inicial.

Todas as superfícies criadas sobre os marcadores definidos pelo usuário e as pontes criadas entre eles são usadas como polígonos no NavMesh, permitindo que os inimigos caminhem livremente em toda a área. Apesar do Unity 3D já fornece uma implementação do NavMesh, a implementação atual funciona apenas com cenários estáticos. Foi necessário utilizar uma implementação disponibilizada pela equipe do Unity que ainda não está disponível na versão utilizada nesse projeto, essa implementação fornece scripts que podem ser inseridos nas superfícies e plataformas para criar polígonos na forma desses objetos em tempo de execução. Além disso foi necessário configurar o NavMesh nos inimigos para eles se tornarem capazes de andar nos polígonos criados, o componente NavMeshAgent é adicionado na criação de todos os inimigos, como o Unity representa os agentes do NavMesh com cilindros, foi necessário especificar o diâmetro dos inimigos, esse valor é utilizado para que os inimigos não alcancem a borda das superfícies, prevenindo eles de caírem tanto das superfícies como das plataformas.

4.6 Resultados

As imagens abaixo exibem o protótipo do jogo de defesa de torres desenvolvido neste trabalho.

A Figura 16 mostra uma visão geral do protótipo desenvolvido, realizando o rastreamento de dois marcadores de imagem e exibindo as plataformas usadas no jogo de defesa de torres. Além das superfícies, as plataformas virtuais geradas pelo algoritmo de Prim também estão em cena. Apenas duas estão sendo exibidas, pois o algoritmo detectou que a distância das outras superfícies estão abaixo do limiar definido, logo os inimigos conseguem pular essa distância sem o auxilio de plataformas virtuais. Nessa figura além das superfícies das plataformas virtuais, o algoritmo detectou as superfícies mais distantes e definiu elas como os pontos de origem e destino dos inimigos. Deste modo os inimigos serão inicializados na superfície mais à esquerda e eles tem como objetivo a superfície mais à direita, que está com um cilindro para representar o ponto final dos inimigos.

A Figura 17 mostra o NavMesh gerado pelo Unity nas superfícies e plataformas virtuais criadas pelo rastreamento e algoritmo de Prim. Como as superfícies são retangulares o NavMesh consegue gerar a malha de polígonos sem problemas, apenas criando um polígono interno e deixando uma borda em cada superfície para que os inimigos não se aproximem delas ou ultrapassem ela. O tamanho da borda é definido pelo raio do cilindro que contém os inimigos. O NavMesh se torna mais complexo nas conexões entre as superfícies pelas

Figura 16 – Visão geral do protótipo de defesa de torres. Fonte: O Autor

plataformas, pois são criados vértices para informar ao algoritmo de busca de caminho qual é o vértice mais próximo.

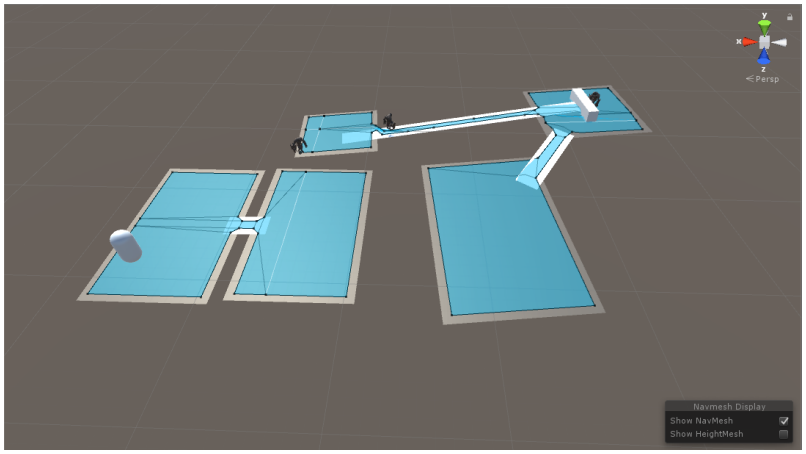

Figura 17 – Navmesh das superfícies e plataformas. Fonte: O Autor

Na Figura 18 temos o marcador de cilindro sendo utilizado para adicionar a torre de defesa que atacará os inimigos que aparecerem no campo de visão da torre. Ao detectar o marcador a torre apenas será ativada se o jogador tiver dinheiro suficiente para o processo, o jogador acumula moedas ao eliminar inimigos.

Os inimigos são inicializados na superfície mais distante do ponto de destino, a Figura 19 mostra um inimigo sendo inicializado na superfície mais à esquerda enquanto alguns outros caminham em direção da última superfície pelas plataformas virtuais. O jogo também cria proceduralmente alguns obstáculos dentro das superfícies obrigando os inimigos desviarem dele e ficando mais tempo na mira das torres. Esses obstáculos também são gerados nas plataformas virtuais, provocando o desativamento temporário

Figura 18 – Visão geral do protótipo com o marcador da torre. Fonte: O Autor

da plataforma, obrigando os inimigos desviarem por outras plataformas. Todos esses obstáculos, tanto das superfícies como das plataformas atualizam o NavMesh, criando novos vértices e polígonos caso necessário.

Figura 19 – Inimigos e obstáculos. Fonte: O Autor

Na Figura 20 temos os marcadores distribuídos de uma maneira diferente das outras imagens. Essa nova disposição dos marcadores gerou um novo grafo e os vértices foram conectados de uma maneira diferente.

4.7 Discussão

Nas seções anteriores foi demostrado como o estudo de caso foi implementado, utilizando os conceitos definidos no capítulo anterior. Algumas dificuldades foram encontradas durante o desenvolvimento. A principal delas foi no rastreamento dos marcadores,

Figura 20 – Nova distribuição dos marcadores. Fonte: O Autor

visto que o Vuforia apresentou algumas dificuldades para reconhecer marcadores com uma distância maior que 60 centímetros, por isso a implementação foi limitada e todo o rastreamento foi realizado em apenas uma mesa e não com vários móveis como planejado.

O estudo de caso desenvolvido aplica todos os conceitos visto nesse trabalho. Apesar da limitação do rastreamento, uma representação foi criada para simular o cenário desejado, distribuindo os marcadores sobre a mesa e criando uma superfície navegável sobre o marcador, simulando o cenário do jogador ao distribuir vários marcadores por diferentes móveis. As plataformas virtuais foram criadas em tempo de execução, gerando um conteúdo procedural e permitindo ao algoritmo se adaptar a qualquer distribuição dos marcadores definidos pelo usuário.

A RA também foi aplicada no reconhecimento dos marcadores. Existe uma limitação do tamanho das superfícies geradas, pois como cada marcador representa uma superfície real do ambiente do usuário, o protótipo deve saber o tamanho do objeto real para gerar uma superfície que ocupe todo o espaço.

5 Conclusão e Trabalhos Futuros

5.1 Conclusão

Esse trabalho apresenta um estudo sobre a criação de jogos com conteúdos procedurais para RA. Vimos no Capítulo 2 que existem poucos estudos sobre esse cenário e que é uma área que ainda pode ser muito estudada e expandida. As ferramentas mostradas no Capítulo 4 são suficiente para desenvolver boas soluções. O Unity é uma ótima ferramenta para desenvolver jogos em RA que fornece uma enorme quantidade de recursos de forma gratuita e o Vuforia se mostrou uma boa solução porém alternativas podem ser estudadas. A qualidade gráfica da aplicação não foi um fator relevante no desenvolvimento da aplicação, utilizando apenas alguns modelos 3D disponíveis na Internet.

O artigo (AZAD et al., 2016) é um dos poucos trabalhos que utilizam conteúdo procedural com RA, porém utilizando o Kinect para rastrear o ambiente e criando dois jogos de protótipo para demonstrar a solução, um jogo baseado no Mario e outro no Lemmings. O jogo da Young Conker (MICROSOFT, 2016) desenvolvido para o HoloLens é a único projeto que utiliza uma mecânica similar e está disponível no mercado, onde o personagem é mostrado no ambiente do jogador e consegue interagir com paredes e móveis, criando elementos virtuais como moedas e plataformas.

Esse trabalho mostra novas alternativas de como é possível utilizar técnicas de PCG e RA em jogos digitais, o protótipo de defesa de torres desenvolvido foi escolhido para ser mais uma opção para futuros trabalhos, mostrando que é possível criar jogos em cenários com RA utilizando uma forma simples de rastreamento como os marcadores, tanto marcadores simples para representar as superfícies como marcadores de multialvos que representaram as torres e sem a necessidade de dispositivos específicos como o Kinect ou HoloLens. O NavMesh do Unity se mostrou capaz de criar uma malha de polígonos para cenários criados em tempo de execução, permitindo a navegação dos inimigos nas superfícies e plataformas criadas.

A solução apresentada mostra que as tecnologias utilizadas nesse trabalho (PCG e RA) se mostram com grande potencial para o desenvolvimento de vários outros tipos de jogos ou aplicações de contexto gerais. O levantamento bibliográfico mostrou alguns trabalhos que utilizam PCG em diversos cenários, como jogos de FPS, ambientes virtuais e até mesmo jogos totalmente procedurais. Vários desses cenários podem ser aplicados em RA para expandir a experiência do usuário e flexibilizar o algoritmo de PCG escolhido.

5.2 Trabalhos Futuros

Existem diversas oportunidades para serem exploradas após esse trabalho, como a criação de novos tipos de jogos utilizando a mesma tecnologia e ferramentas, ou explorando mais a inteligência artificial podendo identificar o ambiente que o jogador está inserido e gerar conteúdos procedurais diferentes para cada tipo de ambiente.

Realizar o rastreamento do ambiente de maneira diferente também pode ser interessante, visto que problemas foram encontrados com o marcador de imagem. O ARKit (APPLE, 2018) vem com uma solução interessante para rastreamento de superfícies sem marcadores. Apesar de ser uma versão inicial já vem mostrando bons resultados que podem ajudar no cenário de desenvolver jogos com RA sem marcadores.

Outro ponto interessante é a visualização de jogos criados com RA com um HMD como o Microsoft HoloLens, onde a imersão fornecida por tais dispositivos podem ajudar no engajamento dos jogos.

Referências

2K. *2K*. 2017. [Online; Acessado em 03 de julho de 2017]. Disponível em: <https://www.civilization.com/>. 15

ANALYSTS, G. I. *Global Industry Analysts*. 2017. [Online; Acessado em 30 de junho de 2017]. Disponível em: <http://www.strategyr.com/MarketResearch/Mobile_Augmented_Reality_MAR_Market_Trends.asp>. 9

APPLE. *ARKit*. 2018. [Online; Acessado em 30 de janeiro de 2018]. Disponível em: <https://developer.apple.com/arkit/>. 35

APTIV8. *Augmented Reality (AR) with Mobile Application and Virtual Reality (VR) Solution*. 2018. [Online; Acessado em 12 de janeiro de 2018]. Disponível em: <https://aptiv8.com/ar-log-vr-solutions/>. 17

ARTOOLKIT. *ARToolkit*. 2017. [Online; Acessado em 02 de julho de 2017]. Disponível em: <https://www.artoolkit.org/>. 10

AZAD, S. et al. Mixed Reality Meets Procedural Content Generation in Video Games. *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, p. 22–26, 2016. 9, 10, 11, 12, 15, 16, 20, 21, 34

AZUMA, R. T. A Survey of Augmented Reality. p. 355–385, 1997. ISSN 18734316. 9, 18

BLIZZARD. *Blizzard*. 2017. [Online; Acessado em 03 de julho de 2017]. Disponível em: <http://us.blizzard.com/pt-br/games/d2/>. 15

CARDAMONE, L. et al. Evolving interesting maps for a first person shooter. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 6624 LNCS, n. PART 1, p. 63–72, 2011. ISSN 03029743. 14

COELHO, A. F. *Procedural Content Generation in Serious Games*. 2015. [Online; Acessado em 12 de janeiro de 2018]. Disponível em: <https://blog.eai.eu/procedural-content-generation-in-serious-games/>. 22

COMPTON, K.; MATEAS, M. Procedural Level Design for Platform Games. *Artificial Intelligence for Interactive Digital Entertainment*, p. 109–111, 2006. Disponível em: <http://www.aaai.org/Papers/AIIDE/2006/AIIDE06-022.pdf>. 15

CONTRIBUTORS, W. *QR code — Wikipedia, The Free Encyclopedia*. 2018. [Online; Acessado em 13 de janeiro de 2018]. Disponível em: <https://en.wikipedia.org/w/index.php?title=QR_code&oldid=818900157>. 19

COOK, M. Would You Look at That! Vision-Driven Procedural Level Design. *Proceedings of the AIIDE Workshop on Experimental AI and Games*, p. 9–14, 2015. 11, 13

COOK, M.; COLTON, S. Multi-faceted evolution of simple arcade games. *2011 IEEE Conference on Computational Intelligence and Games, CIG 2011*, p. 289–296, 2011. 14

CUI, X.; SHI, H. A * -based Pathfinding in Modern Computer A * -based Pathfinding in Modern Computer Games. *IJCSNS International Journal of Computer Science and Network Security*, VOL.11 No., n. March, 2015. 10, 24

Dos Santos, A. B.; DOURADO, J. B.; BEZERRA, A. ARToolkit and Qualcomm Vuforia: An Analytical Collation. *Proceedings - 18th Symposium on Virtual and Augmented Reality, SVR 2016*, p. 229–233, 2016. 10

ENGEL, J.; STURM, J.; CREMERS, D. Semi-dense visual odometry for a monocular camera. *Proceedings of the IEEE International Conference on Computer Vision*, p. 1449–1456, 2013. ISSN 1550-5499. 10

EPPSTEIN, D. *ICS 161: Design and Analysis of Algorithms Lecture notes for February 6, 1996*. 1996. [Online; Acessado em 07 de janeiro de 2018]. Disponível em: <https://www.ics.uci.edu/~eppstein/161/960206.html>. 24

GEIGER, A.; ZIEGLER, J.; STILLER, C. StereoScan: Dense 3d reconstruction in real-time. *IEEE Intelligent Vehicles Symposium, Proceedings*, n. Iv, p. 963–968, 2011. ISSN 1931-0587. 10

GOESELE, M.; STUERZLINGER, W. Semantic Constraints for Scene Manipulation. *Spring Conference in Computer Graphics '99*, p. 140–146, 1999. 13

GOOGLE. *Google*. 2017. [Online; Acessado em 30 de junho de 2017]. Disponível em: <https://developers.google.com/glass/distribute/glass-at-work>. 9

GRIXTI, S. *NO MAN'S SKY MIGHT FINALLY BE GETTING MULTI-PLAYER EXPLORATION*. 2017. [Online; Acessado em 12 de janeiro de 2018]. Disponível em: <https://press-start.com.au/news/playstation/2017/08/11/no-mans-sky-might-finally-getting-multiplayer-exploration/>. 23

HACKEREARTH. *Minimum Spanning Tree*. 2018. [Online; Acessado em 12 de janeiro de 2018]. Disponível em: <https://www.hackerearth.com/practice/algorithms/graphs/minimum-spanning-tree/tutorial/>. 26

HASSABALLAH, M.; MURAKAMI, K.; IDO, S. Face detection evaluation: A new approach based on the golden ratio. v. 7, 03 2013. 20, 21

HJELM, E.; HJELMÅS, E.; LOW, B. K. Face Detection: A Survey. v. 83, n. 3, p. 236–274, 2001. ISSN 10773142. 20

HORSWILL, I.; FOGED, L. Fast Procedural Level Population with Playability Constraints. *8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2012)*, p. 20–25, 2012. Disponível em: <http://aaai.org/ocs/index.php/AIIDE/AIIDE12/paper/view/5466/5691>. 15

IRONHIDE. *Kingdom Rush*. 2011. [Online; Acessado em 04 de março de 2018]. Disponível em: <https://www.ironhidegames.com/Games/kingdom-rush>. 28

MICROSOFT. *Young Conker*. 2016. [Online; Acessado em 07 de janeiro de 2018]. Disponível em: <https://www.microsoft.com/en-au/hololens/apps/young-conker>. 34

MICROSOFT. *Microsoft HoloLens*. 2017. [Online; Acessado em 30 de junho de 2017]. Disponível em: <https://www.microsoft.com/en-us/hololens>. 9

MICROSOFT. *Microsoft Kinect*. 2017. [Online; Acessado em 02 de julho de 2017]. Disponível em: <https://developer.microsoft.com/pt-br/windows/kinect>. 9

MILGRAM HARUO TAKEMURA, A. U. F. K. P. *Augmented reality: a class of displays on the reality-virtuality continuum*. 1995. 2351 - 2351 - 11 p. Disponível em: <http://dx.doi.org/10.1117/12.197321>. 18

MOHAMMADKHANI, M. Kinect 3D Reconstruction. *Mahsam.Ir*, 2013. Disponível em: <http://mahsam.ir/615/Mahsa-Mohammadkhani-Final-Report.pdf>. 10

NEWZOO. *Newzoo*. 2017. [Online; Acessado em 30 de junho de 2017]. Disponível em: <https://newzoo.com/insights/articles/the-global-games-market-will-reach-108-9-billion-in-2017-with-mobile-taking-42/>. 9

NINTENDO. *Mario*. 2017. [Online; Acessado em 03 de julho de 2017]. Disponível em: <http://mario.nintendo.com/>. 15

OCULUS. *Oculus Rift*. 2017. [Online; Acessado em 30 de junho de 2017]. Disponível em: <https://www.oculus.com/rift/>. 9

SHINDE, J. *How Apple, Google, Microsoft and Facebook Are Trying To Augment Our Reality Like Never Before*. 2017. [Online; Acessado em 30 de janeiro de 2018]. Disponível em: <https://www.indiatimes.com/technology/news/how-apple-google-microsoft-facebook-are-trying-to-augment-our-reality-like-never-before-329742.html>. 20

SUBATOMIC. *Fieldrunners*. 2008. [Online; Acessado em 04 de março de 2018]. Disponível em: <http://subatomicstudios.com/games/fieldrunners/>. 28

TOGELIUS, J. et al. Search-Based Procedural Content Generation : A Taxonomy and Survey. v. 3, n. 3, p. 172–186, 2011. 9, 13

TSUBAKI. *How to use NavMesh part 2*. 2012. [Online; Acessado em 12 de janeiro de 2018]. Disponível em: <http://tsubakit1.hateblo.jp/entry/20120128/1327679742>. 28

TUTENEL, T.; SMELIK, R. M. Using Semantics to Improve the Design of Game Worlds. *Fifth Artificial Intelligence for Interactive Digital Entertainment Conference*, n. September 2015, p. 100–105, 2009. 11, 13

UNITY. *Unity 3D*. 2018. [Online; Acessado em 07 de janeiro de 2018]. Disponível em: <https://unity3d.com/pt/>. 24

VUFORIA. *Vuforia*. 2017. [Online; Acessado em 02 de julho de 2017]. Disponível em: <https://developer.vuforia.com/>. 10, 24, 25

WIKIPéDIA. *Lemmings (jogo eletrônico) — Wikipédia, a enciclopédia livre*. 2018. [Online; Acessado em 7 de janeiro de 2018]. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Lemmings_(jogo_eletr%C3%B4nico)&oldid=43476244>. 15