Shambles

João Azevedo*
N. Mec: 201200316
FCUP-DCC

José Paiva†
N. Mec: 201200272
FCUP-DCC

Ricardo Rocha‡
N. Mec: 201003917
FCUP-DCC

Figura 1: *Duas imagens do Shambles em zonas distintas do mapa - à esquerda numa zona central em cima de uma ponte e à direita no lago - mostrando a eficiência dos NPCs a seguir o jogador. (nota: imagens clareadas para efeitos de impressão)*

Resumo

Shambles é um jogo de terror, first-person shooter, desenvolvido em Unity 3D [Unity3D 2014] , que leva a personagem (o jogador) a um mundo pós-apocalíptico, rodeado de mortos-vivos. O objectivo da personagem é acabar com esta praga, matando todos os *zombies* que encontrar, antes que estes o matem. Shambles conta com um cenário em que a superfície terrestre está coberta de neve, e onde o único humano vivo é o jogador, e todos os outros *habitantes* devem ser exterminados. No Shambles o jogador dispõe de diversas armas para que possa exterminar todos os mortos-vivos, tais como um revólver, uma caçadeira, uma metralhadora, uma arma lançadora de chamas, um lançador de misseis, entre outras.

Shambles possui ainda um ranking Online, de forma a que os jogadores possam competir, lutando assim por se tornarem o melhor jogador do mundo de Shambles.

Keywords: unity, first-person shooter, zombies, mortos-vivos

Links: VIDEO

1 Introdução

O mundo dos videojogos, e neste caso concreto, o mercado dos Fisrt-Person Shooter é hoje em dia muito lucrativo. Em 2013 o mercado dos FPS contava com uma receita de 1,4 biliões de dólares (cerca de 1 bilião de euros) [ENGBER 2014].

A data de lançamento do primeiro FPS remonta para 1991, data em que foi lançado o jogo Catacomb 3D. Este jogo foi criado apenas por 6 pessoas [Wikipedia 2014], e era um jogo de magia em que o jogador só via as mãos do personagem, introduzindo assim a característica principal dos FPS.

Em 2013 foi lançado Crysis 3 que conta com um custo de produção de 66 milhões de dólares (48,4 milhões de euros), sendo um dos jogos mais caros até aqui criados [WIKI 2014]. Comparando os dois jogos mencionados é possível ver o quão os gráficos melhoraram, as imensas novas *features* que foram acrescentadas, como modo Online, os novos tipos de controlo e a melhoria da qualidade sonora, por exemplo.

Outra referência é o jogo Crysis 2 datado de 2011 que precisou de cerca de 200 pessoas para ser desenvolvido, isto sem contar com os tester's. Este demorou ainda 2 anos e meio a ser desenvolvido e o seu desenvolvimento ocorreu em várias localizações [Takahashi 2011].

Shambles por sua vez é um jogo em Unity 3D, desenvolvido em 3 meses, e que ilustra o potencial do Unity, já que neste curto espaço de tempo (comparado a outros jogos do género) se conseguiu criar um jogo visualmente apelativo e com uma jogabilidade próxima dos First-Person Shooter's que podemos encontrar nas lojas. Shambles está disponível para Windows 7+, Linux e Mac, e conta também com uma versão para Webplayer. O jogador poderá jogar com teclado e rato ou comando.

2 Estado da Arte / Trabalhos Relacionados

Dentro da área dos jogos de terror com *zombies* existem já alguns exemplos de first-person shooters bem sucedidos, como é o caso do *Left 4 Dead*, *Resident Evil* ou *Killing Floor*, sendo que este último também é um *Survival*.

Em relação a plataformas para desenvolvimento de jogos, existem atualmente diversas, como o *Unity* já mencionado, o *GameMaker: Studio* [Games 2014], *Unreal Engine 3*, [Engine 2014] entre outros. Estas ferramentas de desenvolvimento são apreciadas pois permitem criar jogos sem necessitar de ter orçamentos elevadíssimos, e sem ser necessárias equipas com tantos elementos como por exemplo a do *Crysis 2*. A escolha de uma destas plataformas é dependente do tipo de jogo a desenvolver sendo que, por exemplo o *Unity* permite o desenvolvimento da maioria dos tipos de jogos em 2D ou 3D. Para o Shambles foi escolhido como já referido o *Unity* pois para além de tudo isto, é de fácil aprendizagem e intuitivo.

*e-mail: up201200316@fc.up.pt
†e-mail: up201200272@fc.up.pt
‡e-mail: up201003917@fc.up.pt

3 Ferramentas Utilizadas

O Unity [Unity3D 2014] é uma plataforma para desenvolvimento de jogos na qual são incorporadas várias ferramentas fundamentais na construção de um jogo, facilitando assim o trabalho aos programadores e tornando mais rápido o processo. Existem já milhares de jogos feitos nesta plataforma, e outros tantos plugins e frameworks, como o Terrain Toolkit (agora descontinuado) e o FPS Constructor [DastardlyBanana 2009] utilizados no Shambles, próprios de cada categoria de jogo.

O FPS Constructor [DastardlyBanana 2009] contém já modelos de armas e algoritmos para controlar uma personagem de um first-person shooter nos quais é necessário apenas ligeiras alterações para termos a personagem a funcionar no nosso jogo. Esta framework foi usada no processo de desenvolvimento do Shambles, assim como modelos 3D inanimados presentes na biblioteca do Unity e sons gratuitos [SoundBible 2006].

4 Trabalho Desenvolvido

Shambles é um jogo realizado em Unity 3D, onde é possível realizar várias acções com o personagem, tais como matar *zombies*, caminhar, correr, baixar, rastejar, inclinar, saltar e apanhar armas.

Foram também implementadas várias técnicas de computação gráfica, que descrevemos seguidamente, juntamente com as várias etapas do projecto.

4.1 Game Design

4.1.1 Gameplay

O personagem vê-se no meio de um apocalipse *zombie*, e o seu objectivo é sobreviver o máximo de tempo possível matando o maior número de *zombies* com as armas que apanhar do chão e vida e balas que conseguir de bónus dos *zombies* que matar.

4.1.2 Menu

O menu (2) construído para o Shambles foi feito com a finalidade de apresentar medo ao jogador desde o primeiro contacto com o jogo. O menu está assente numa imagem com imenso sangue antevendo o que o jogador irá encontrar no jogo. Pode-se também encontrar ossos a voar pelo ecrã servindo de presságio ao jogador para o que lhe espera se não tiver cuidado no Shambles. O menu conta ainda com efeitos sonoros de forma a vincar a sensação de medo.

Figura 2: *Menu Principal do Shambles*

Para além do menu principal tem um menu de pausa no mesmo estilo acedido clicando o botão *Esc* durante o jogo, ou activado automaticamente quando se muda de aplicação, e também um local para consultar os *Highscores* do jogo.

4.1.3 Terreno/Cena

O terreno deste demo foi construído usando as ferramentas oferecidas pelo Unity para este efeito, a partir de uma planície vazia. Os relevos foram transformados a partir de um *Heightmap*, de seguida aplicaram-se as texturas de neve, em zonas montanhosas, e de rocha, nas menos montanhosas. A vegetação presente no mapa foi criada usando a funcionalidade *Mass Place Trees* (colocação de árvores em massa), e a partir do resultado obtido foram eliminadas as árvores que ficaram em sítios indesejáveis.
Escolhemos uma *Skybox* - um contentor de toda a cena que mostra um mundo vasto para lá do terreno - repleta de nuvens e implementamos uma pequena densidade de nevoeiro à altura do topo das árvore e a uma distância mediana do personagem.
Foram utilizados modelos 3D de uma cabana, uma ponte, um *bunker*, um avião destruído e rochas, presentes na *Unity Asset Store*, para tornar o mapa mais realista e atrativo.

4.1.4 Estética

O Shambles, sendo um jogo pós-apocalíptico, deve transmitir ao jogador algum factor medo durante o jogo. A iluminação do mapa é crucial para isso, por essa razão demos apenas uma pequena iluminação (3) ao jogador, vinda do ”sol”. Esta é uma luz direcional com intensidade 0.3 e cor cinzento-escuro que faz o mapa tornar-se bastante escuro.

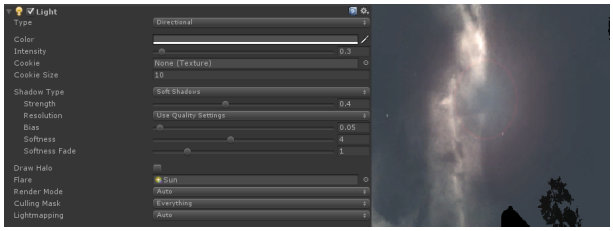


Figura 3: *Propriedades da luz e resultado*

As sombras das árvores emitidas por esta luz contrastam pouco com o resto do chão devido à pouca luminosidade, dando a ideia temporal de final de tarde / inicio de noite.

Com o mapa tão escuro notou-se a necessidade de uma luz no *Bunker* para o jogador ver os *pickups* lá existentes, que poderia usar. Para isso foi colocado um pequeno candeeiro no tecto do *Bunker* com a iluminação adequada para uma pequena luz interior já velha.

4.1.5 Windzone

O vento é outro factor associado ao terror, que torna um jogo muito mais realista e assustador. Neste jogo criamos uma *Windzone* direcional (4) que provoca efeitos de vento, reais, nas árvores colocadas no terreno. Com esta fonte colocou-se também som de vento forte, por todo o mapa. [Unity3D 2011]

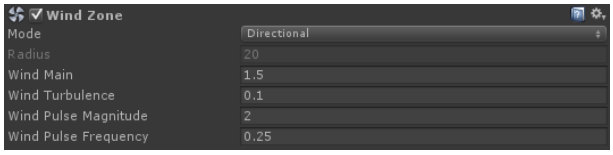


Figura 4: *Propriedades do vento: Wind main - força do vento, Turbulence - turbulência na força do vento, Pulse Magnitude - variação do vento ao longo do tempo, Pulse Frequency - frequência das mudanças do vento*

4.1.6 Queda de neve

Devido ao terreno conter neve criou-se um sistema de partículas para emitir neve por todo o mapa. Este sistema é constituído por um emissor com o tamanho do mapa colocado muito acima deste, e com a particularidade de emitir as partículas na direção do vento de forma a torná-lo mais realista.

4.1.7 Sons

Começando pelo menu temos uma música de fundo típica dos filmes de terror, e em cada item temos um grito de um *zombie* quando o rato passa por cima. Também na zona dos *highscores* existe uma música de fundo.

No decorrer do jogo existem vários sons chamados dependentes das acções feitas pelo utilizador, entre os quais som dos tiros de cada arma, pegadas nas diferentes estruturas - neve, água, metal, madeira e cimento -, do embate dos tiros, do ataque e respiração ofegante dos *zombies*, e quando o personagem é atingido. Para além destes sons, existe constantemente o som do vento forte e do resultante abanar das árvores.

Foi também definida as prioridades dos sons, sendo que o vento tem menor prioridade do que os outros sons, seguido das pegadas, respiração e ataque dos *zombies* (depende da proximidade), e dos tiros. [Unity3D 2013a]

4.2 Game Development

4.2.1 Navigation Mesh, Pathfinding e Target Following

Os terrenos presentes no Shambles podem ser bastante irregulares e, por isso é necessário usar uma rede com possíveis caminhos para os NPCs percorrerem, para que não fiquem a caminhar contra uma montanha que não dá para subir, por exemplo. Para isso foi usada uma navmesh, que é uma estrutura de dados abstrata (grafo) usada para implementar inteligência artificial, na procura de caminhos em espaços amplos e com obstáculos. A navmesh representa os caminhos calculados pelo algoritmo de Pathfinding, e durante a execução do jogo escolhe-se um dos caminhos possíveis entre cada dois pontos.

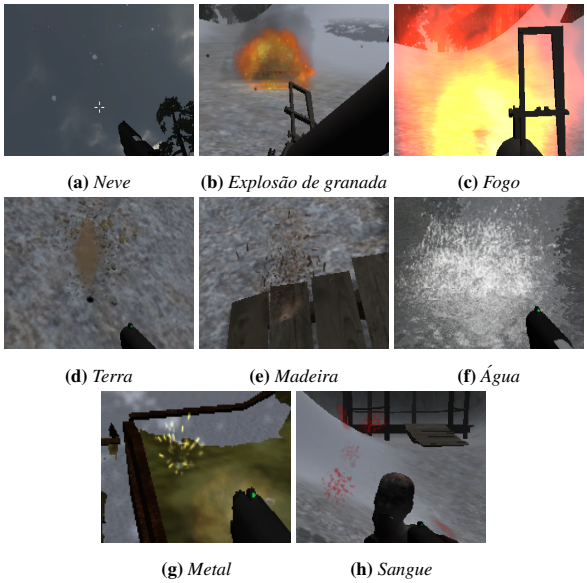
No Shambles foi usado o gerador de navmesh presente no Unity para criar uma representação da área que os *zombies* podem percorrer, visto que é aplicada a técnica de Target Following - seguir um alvo, neste caso o jogador - e o caminho por estes feito nunca é o mesmo e em cada nova frame é preciso recalcular o caminho do NPC para o personagem. Como existem vários algoritmos com baixa complexidade para a procura de caminhos / caminhos mais curtos em grafos, a navmesh é uma solução viável para usar nos NPCs de um jogo .

4.2.2 Sistemas de Partículas

Um sistema de partículas é uma técnica de computação gráfica que usa um grande número de pequenos objectos gráficos para simular fenómenos como explosões, chuva, água a saltar, entre outras coisas.

O Shambles faz uso desta técnica para criar efeitos como o da neve a caír (5a), a explosão de granadas (5b), o fogo do lançador de chamas (5c), os tiros nos diversos tipos de objectos - madeira (5e), metal (5g), terra (5d) e água (5f) - e sangue (5h). Alguns destes sistemas de partículas pertencem ao Unity e foram apenas adequados e invocados nas diversas situações, outros como o sangue e a neve foram alterações de sistemas lá existentes. O fogo do lançador de chamas (5c) e a explosão da granada (5b) foram invocados e alterados do FPS Constructor.

Cada sistema tem a textura apropriada do objecto a emitir, assim como valores (6) de emissão (varia entre os 150-200/frame para a neve, e os 10-50/frame para os restantes), velocidade e energia. [Unity3D 2013b]



(a) *Neve* **(b)** *Explosão de granada* **(c)** *Fogo*

(d) *Terra* **(e)** *Madeira* **(f)** *Água*

(g) *Metal* **(h)** *Sangue*

Figura 5: *Sistemas de partículas*

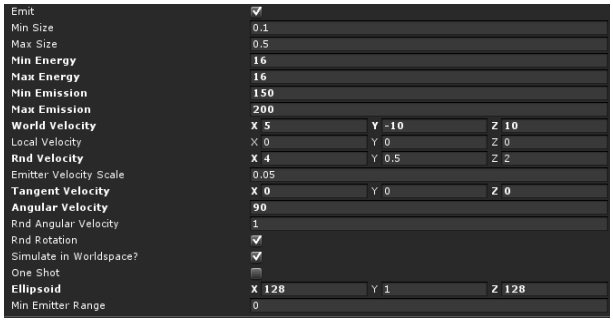


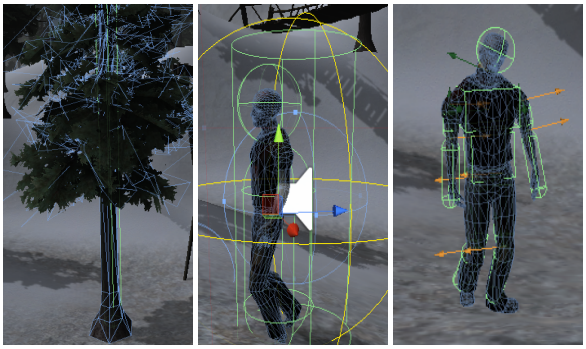
Figura 6: *Exemplo de um sistema de partículas em Unity 3D, neste caso o da neve*

4.2.3 Sistemas de Colisões

A detecção de colisões de um jogo é uma parte crucial para o realismo e a funcionalidade do jogo, já que são as colisões que permitem ao jogador interagir com o mundo virtual. No Unity a detecção é feita a partir de uma rede de polígonos ou objectos básicos como esferas, caixas e cápsulas, chamados *Colliders*, sendo que um modelo pode ser constituído por dezenas de *Colliders*.

No Shambles temos *Colliders* nos modelos de infraestruturas importados da Unity Asset Store que usam uma rede de polígonos, e noutros modelos como é o caso dos *zombies*, árvores, personagem, modelos coletáveis (balas, armas e primeiros socorros) e água implementou-se sistemas de colisões à base dos *Colliders* mais simples. Com os *Triggers* dos vários *Colliders* invocam-se os sistemas de partículas, sons e funções de danos.

Para as árvores (7a) foi criado uma cápsula apenas no tronco, impedindo o jogador de passar pelo meio.
Na água foi colocado uma caixa com altura 0 (zero) na superfície, para podermos invocar o sistema de partículas quando as balas colidem e outra no fundo da água para activar os sons do caminhar do jogador, visto que a colisão entre o jogador e a superfície foi desactivada para permitir a entrada na água.
Os *zombies* (7b), sendo um modelo complexo e precisando de detectar com precisão os tiros recebidos, são constítuidos por vários *Colliders*, esfera (cabeça), cápsula (tronco e pernas) e cilindro (corpo inteiro). Existem também sistemas de colisões diferentes para os *Rigidbodies* (7c) que substituem os *zombies* "vivos", que são constituídos por cápsulas nos membros, no tronco uma caixa e na cabeça uma esfera.
O sistema de colisões da personagem/jogador é apenas constituído por uma cápsula, sendo que as armas e braços são levantados quando estão muito perto de algum objecto, evitando assim a colisão.
Nos modelos coletáveis é usado apenas uma caixa com o tamanho do modelo.



(a) *Árvores* **(b)** *Zombie activo* **(c)** *Zombie morto*

Figura 7: *Sistema de Colisões, os Colliders encontram-se com a cor verde*

4.2.4 Occlusion Culling

Identificar eficientemente os objectos visíveis do ponto de vista do jogador pode ser um processo bastante caro a nível de processamento. *Occlusion Culling* é uma técnica de computação gráfica com o objectivo de evitar renderização de áreas invisíveis ao utilizador, que estão tapadas por outros objectos do mundo virtual. [Coorg and Teller 1997]

Esta é uma técnica usada neste jogo para evitarmos renderizar objectos por trás de árvores, montanhas e outros, de modo a aumentar a velocidade de processamento do jogo.

4.2.5 Enemy Spawn

Um *spawn* é um objecto com um algoritmo associado que instancia objectos, neste caso um *enemy spawn* cria objectos inimigos ao jogador. No Shambles foram criados cinco objectos deste tipo que criam mortos-vivos, a cada espaço de tempo aleatório entre um e cinquenta segundos, sendo que para não abusar do processamento o algoritmo só cria novos se o número de *zombies* não ultrapassar o limite máximo definido (trinta). O sítio destes objectos no mapa foi escondido o máximo possível para impedir o jogador de ver a instanciação dos objectos, e para além disso implementou-se uma função que só permite o *spawn* nos pontos em que o jogador se encontra a tal distância que torne impossível de ver o *zombie* ser criado.

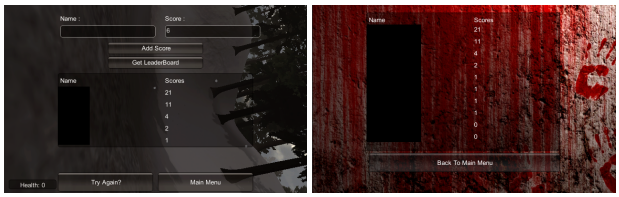
4.2.6 Pickups e Bónus

No mapa existem vários itens espalhados que o jogador pode apanhar usando a tecla *tab* quando a sua direção da visão estiver para lá voltada. Existem três categorias de itens, balas, saco de primeiros socorros (vida) e armas. Também quando se mata um inimigo existe uma componente aleatória que pode dar um bónus de vida ou balas para o jogador apanhar. O sistema de apanhar objectos foi construído com base no existente no FPS Constructor [Dastardly-Banana 2009].

4.2.7 Highscores

A zona de *Highscores* foi construída de modo a que estes estejam disponíveis online, fomentando assim uma competição mais vasta entre os jogadores. Utilizou-se para este efeito uma base de dados num servidor online, ao qual se acede com dois *scripts PHP* (um para consultar os *highscores* já existentes e outro para submeter um novo) chamados por *Javascript* do Unity.

Todas as vezes que o jogador é morto aparece uma interface (8a) que permite a submissão dos pontos e ver os cinco maiores scores. Existe também uma zona acessível a partir do menu inicial para ver os *highscores* (8b). [Lindeman 2013]



(a) *Submissão* **(b)** *Visualização*

Figura 8: *Submissão e visualização de Highscores*

5 Conclusões e Resultados

Neste projecto foi elaborado jogo first-person shooter, de terror, em que o jogador é colocado num ambiente pós-apocalíptico numa zona de baixa temperatura. O terror instalado neste jogo é resultante do som, iluminação e dos NPCs (*zombies*).

5.1 Trabalho Futuro

Os focos de expansão deste projecto deverão incidir sobre o aumento da variedade dos tipos de NPCs, a inserção de novos mapas e alguns melhoramentos sugeridos por *testers*. Principalmente, queremos testar a possibilidade de incluir um sistema de realidade aumentada semelhante ao descrito por Piekarski e Thomas [Piekarski et al. 2003], para o first-person shooter online *Quake*, que na altura em que foi feito ainda não existia tecnologia de imersão como a agora existente (por exemplo os Oculus Rift [OculusVR 2014]), mas provou já ser possível a imersão dos jogadores nos first-person shooter com um *Head Mounted Display*, dispositivos de geo-localização, um portátil e alguns acessórios para os jogadores terem a ideia da arma.

O aumento da imersão dos jogadores em First-Person Shooters tem sido alvo de vários estudos nos últimos anos, como o *Performance*

of modern gaming input devices in first-person shooter target acquisition [Zaranek et al. 2014] que faz uma avaliação da performance de alguns novos dispositivos de input, como o Kinect e o Move, em comparação aos habituais num jogo deste género, e que nos testes que fizeram (com alunos de Universidade) verificaram que a performance era bastante mais baixa com os novos dispositivos. Ainda que a performance não esteja ao mesmo nível acredita-se que se deve à falta de hábito e que é viável um maior nível de imersão em FPSs.

Os testes ao jogo serão realizados na versão demo para *Webplayer* disponível online, a um conjunto de pelo menos vinte pessoas com idades compreendidas entre os 15 e os 30 anos, nos quais será avaliada a reacção destes durante cinco minutos de jogo, e no final será apresentado um pequeno leque de perguntas de resposta rápida para termos uma ideia se irão jogar novamente, sugestões e o nível de medo sentido (será avaliado também por nós durante o teste).

5.2 Limitações

Shambles tem alguns limites no número de NPCs e de partículas emitidas, por razões de processamento. O número de NPCs em campo foi limitado para 30 pois este é o máximo que se consegue atingir num computador com requisitos mínimos (CPU dual-core a 2GHz, gráfica com 256MB e 4GB de RAM) sem que este interfira no normal desempenho do jogador. O número de partículas emitidas por frame não deve também exceder os 200 pelas mesmas razões.

Outra limitação/bug está no facto dos NPCs não se movimentarem exactamente no solo, quando estão em zonas montanhosas. Isto deve-se à animação feita para o estilo de caminhar deles não incluir a inclinação das subidas. Também o *Collider* (cápsula única para as duas pernas 7b) usado nos membros inferiores não é adequado para estas zonas.

Referências

COORG, S., AND TELLER, S. 1997. Real-time occlusion culling for models with large occluders. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, ACM, 83–ff.

DASTARDLYBANANA, 2009. Fps constructor. http://www.dastardlybanana.com/FPSConstructorWeapons.htm.

ENGBER, D., 2014. Who made that first-person-shooter game? http://www.nytimes.com/2014/02/02/magazine/who-made-that-first-person-shooter-game.html?_r=0.

ENGINE, U., 2014. Unreal engine. https://www.unrealengine.com/.

GAMES, Y., 2014. Gamemaker: Studio. https://www.yoyogames.com/studio.

LINDEMAN, K., 2013. Server side highscores. http://wiki.unity3d.com/index.php?title=Server_Side_Highscores.

OCULUSVR, I., 2014. Oculus rift. http://www.oculusvr.com/rift/.

PIEKARSKI, W., THOMAS, B. H., ET AL. 2003. *ARQuake-Modifications and hardware for outdoor augmented reality gaming*. PhD thesis, Linux Australia.

SOUNDBIBLE, 2006. Free sounds. http://soundbible.com/.

TAKAHASHI, D., 2011. Crysis 2 pushes the extreme edge of graphics in video games. http://venturebeat.com/2011/03/22/qa-crysis-2-pushes-the-extreme-edge-of-graphics-in-v

UNITY3D, 2011. Tree - wind zones. http://docs.unity3d.com/Documentation/Components/class-WindZone.html.

UNITY3D, 2013. Audio source. https://docs.unity3d.com/Documentation/Components/class-AudioSource.html.

UNITY3D, 2013. Particle systems overview. https://docs.unity3d.com/Documentation/Manual/ParticleSystems.html.

UNITY3D, 2014. Unity 3d. http://unity3d.com/pt.

WIKI, W. V. G. S., 2014. Most expensive video games? http://vgsales.wikia.com/wiki/Most_expensive_video_games.

WIKIPEDIA, 2014. Catacomb 3-d. http://en.wikipedia.org/wiki/Catacomb_3-D.

ZARANEK, A., RAMOUL, B., YU, H. F., YAO, Y., AND TEATHER, R. J. 2014. Performance of modern gaming input devices in first-person shooter target acquisition. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI EA '14, 1495–1500.