Desenvolvimento com Motores de Jogos II

Aula 10 - Materials - Parte 2

Apresentação

Olá pessoal, sejam bem-vindos à aula 10. Já chegamos em mais da metade da nossa disciplina! Mas ainda há muito a se cumprir, vamos lá! Na aula de hoje, aprenderemos a criar Materials mais avançados, com configurações personalizadas dos parâmetros do Standard Shader. Além das configurações que já vimos, como a do atributo Albedo, a qual determina a cor e a textura, estudaremos os atributos Metallic, Smoothness, Normal Map, Height Map, Occlusion Map e Emission.

Objetivos

Ao final desta aula, você deverá ser capaz de:

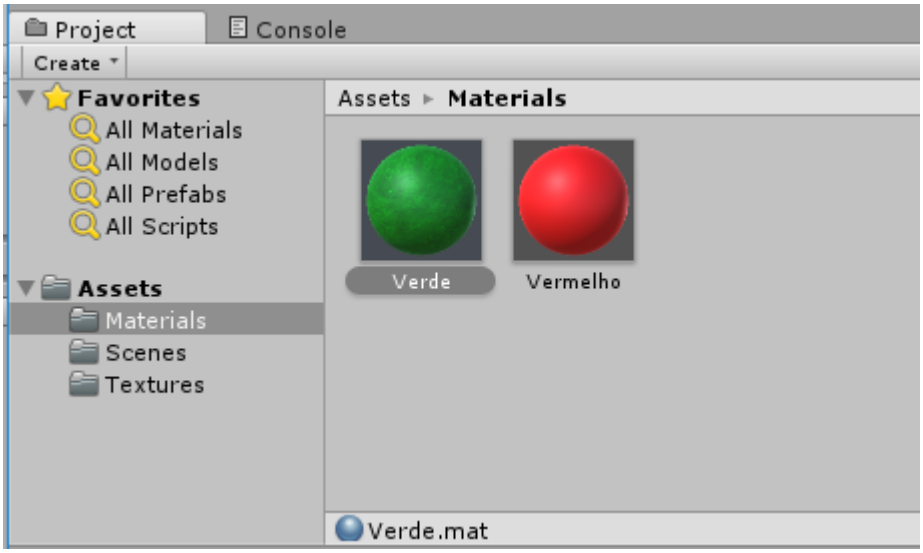
- Criar Materials avançados;
- Aplicar diferentes Materials em objetos.

Parâmetros Avançados do Material

Muitas configurações podem ser feitas em um Material, não somente em sua cor e textura, como já vimos, mas também outras configurações, tais como de luminosidade, de suavização da superfície, das áreas serem mais ou menos suscetíveis a sombras, etc.

No nosso projeto MaterialsTest criamos, até então, dois Materials, um chamado Verde e outro chamado Vermelho. O Material Vermelho é somente uma cor sólida sem configurações extras, e o Material Verde está atualmente configurado com uma textura de grama adicionada na aula passada. Veja na **Figura 1** esses dois Materials na janela Project e observe que o Verde tem um ícone mostrando, para ilustrar, a textura de grama aplicada em uma esfera.

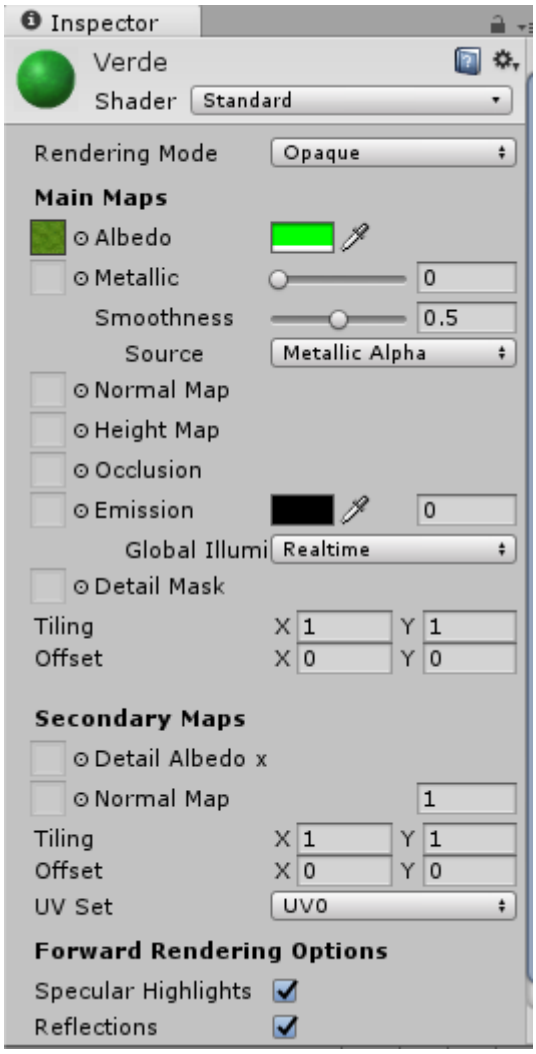
Figura 01 - Material Verde com textura de grama aplicada no seu ícone.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Agora, veremos a descrição e o funcionamento das propriedades mais importantes do Material. Você pode acompanhar a aula com o Material Verde aberto, aplicando e testando cada uma das configurações abordadas. As propriedades atuais do Material Verde estão expostas na **Figura 2** e podem ser acessadas clicando sobre ele e exibindo-as no Inspector.

Figura 02 - Propriedades atuais do Material Verde.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Todas as propriedades são importantes e cada uma tem sua função. Nesta aula, veremos as citadas baixo, que dizem respeito às mais relevantes para o aprendizado inicial de desenvolvimentos de jogos 3D.

Propriedades de desenvolvimento de Jogos 3D:

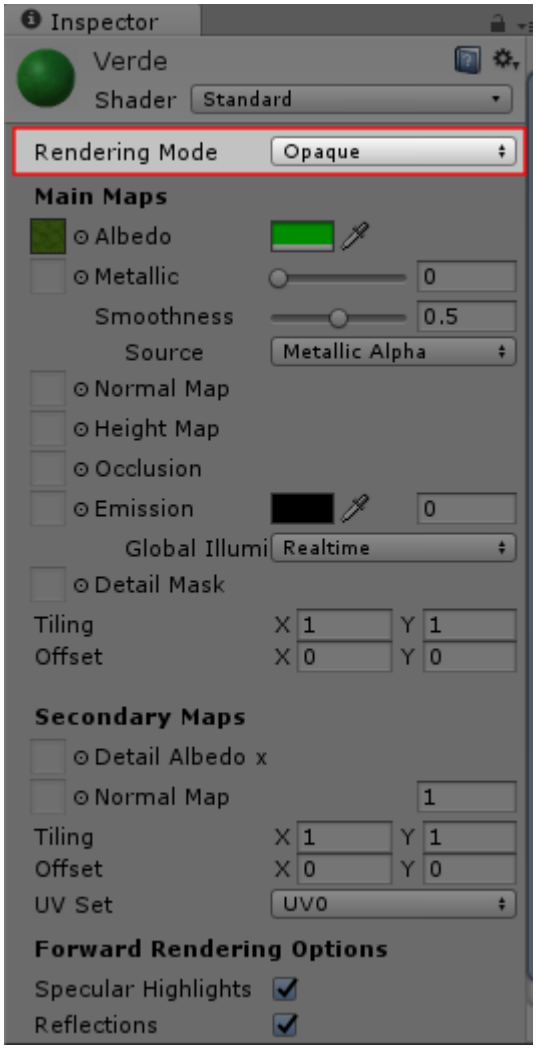
- **Rendering Mode:** configurações de transparência.
- **Albedo:** cor base e textura.
- **Metallic:** refletividade da superfície.
- **Smoothness:** rugosidade da superfície.

- **Normal Map:** mapa de detalhes de superfície.
- **Height Map:** mapa de protuberâncias da superfície.
- **Occlusion Map:** mapa de sombras da superfície.
- **Emission:** emissão de luz da superfície.

Rendering Mode

O **Rendering mode** permite que você determine se o objeto utiliza ou não transparência na sua superfície e, caso utilize, de que tipo ela é. Na **Figura 3**, você vê em destaque essa propriedade.

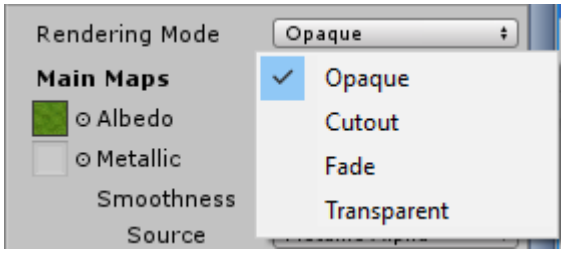
Figura 03 - Rendering Mode.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Os valores que ele pode receber são escolhidos na caixa de seleção, por meio da configuração padrão Opaque (opaco), mas podem ser mudados simplesmente ao clicar nessa caixa e escolher outro. Veja a **Figura 4**.

Figura 04 - Mudando o valor do Rendering Mode.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

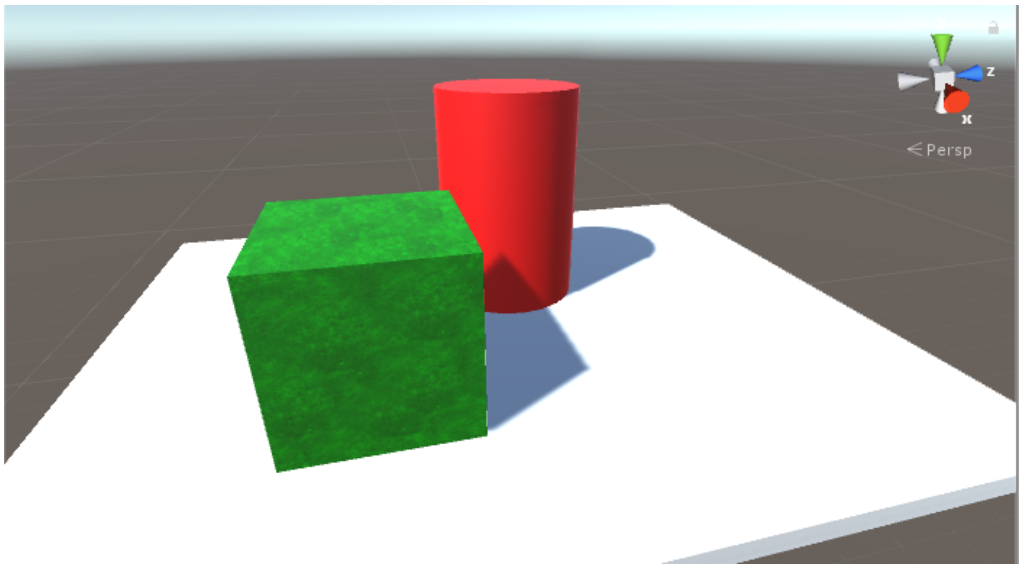
Os valores possíveis são: **Opaque, Cutout, Fade e Transparent**. Eles serão discutidos separadamente a seguir.

Opaque

Esse é o valor padrão e serve para Materials que serão aplicados a objetos nos quais se espera que sua superfície tenha um aspecto sólido, sem transparência.

A **Figura 5** mostra um cubo com o Material Verde (com textura de grama) configurado como **Opaque**. Observe que é exatamente o mesmo efeito visto até então, opaco e sem transparência. Na cena também foi adicionado, no fundo, um Cylinder Vermelho que estará sempre com o Material padrão. Esse Cylinder está somente com a cor sólida vermelha, a qual foi aplicada para efeitos de comparação.

Figura 05 - Material Verde (com texture de grama) aplicado em um cubo com Rendering Mode Opaque.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Cutout

Permite você criar um efeito transparente com transições diretas entre áreas opacas e transparentes. Nesse modo não existe o conceito de “semitransparente”, isto é, ou as áreas da textura são totalmente opacas ou totalmente transparentes. Isso é muito útil quando você está criando uma textura, por exemplo, de uma folha na qual você deseja o contorno determinando que a área interna seja visível e a área externa não. Outro exemplo é uma textura de um tecido perfurado no qual você deseja que as áreas perfuradas sejam totalmente transparentes. Para usar esse modo, você deve escolher uma textura na qual o canal Alpha esteja configurado, ou seja, que tenha informações de quais áreas são transparentes, como essa imagem de uma folha vermelha na **Figura 6.**

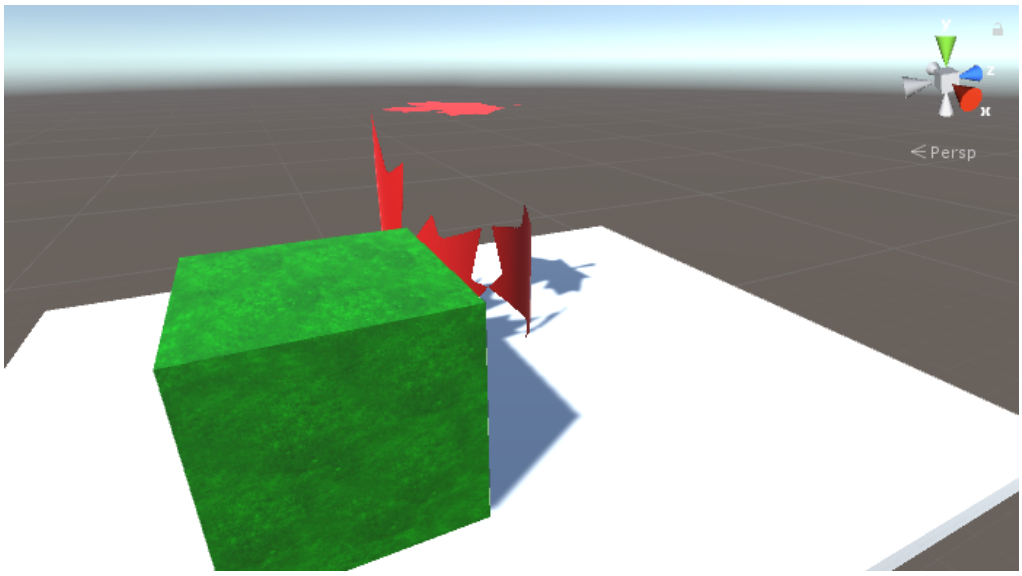
Figura 06 - Textura com transparência que pode ser utilizada no modo Cutout.

Fonte: https://upload.wikimedia.org/300px-Red_Maple_Leaf.svg.png.

Repare que na **Figura 6** existe uma área quadriculada ao redor da folha. Essa área representa a parte transparente dessa imagem que, quando utilizada em um Material, e você deseje que essa área seja também transparente quando aplicada em um objeto, então o Material deve ter o **Rendering Mode** configurado em **Cutout**.

Veja na **Figura 7** a mesma cena com o Material Vermelho configurado com essa textura da folha e com o **Rendering Mode Cutout**.

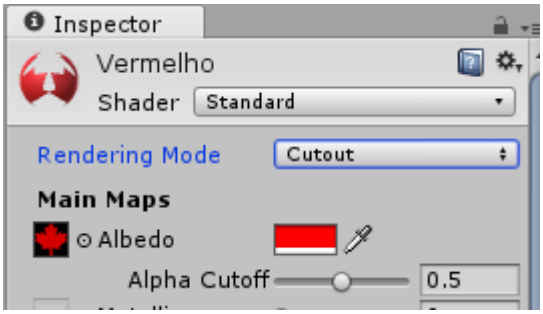
Figura 07 - Material Vermelho com a textura da folha e o Rendering Mode Cutout ativado.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

As configurações do Material Vermelho realizadas foram as mudanças da Textura para essa folha (já importada na pasta Textures) e o **Rendering Mode** para **Cutout**. Veja a **Figura 8**.

Figura 08 - Configuração do Material Vermelho no modo Cutout e com uma textura com transparência.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Repare que uma nova propriedade aparece na configuração do Material Vermelho quando escolhemos o modo **Cutout**. Essa configuração é o **Alpha Cutoff**, o qual pode variar de 0 a 1. Ela é necessária, pois o canal Alpha de transparência de uma imagem não é simplesmente binário dessa forma (transparente ou não), ou seja, ele determina qual o grau de transparência de cada área da imagem.

No modo **Cutout** isso não é suportado, ou seja, cada pixel da imagem é 100% transparente ou 100% opaco. Então, a configuração **Alpha Cutoff** determina o nível de tolerância do canal Alpha da textura, convertendo os valores intermediários de transparência para 100% ou 0% transparentes, de acordo com a tolerância escolhida no **Alpha Cutoff**. Essa tolerância, por padrão, é 0.5, ou seja, tudo o que for mais do que 50% transparente é considerado 100%, e tudo o que for abaixo disso é considerado 100% opaco.

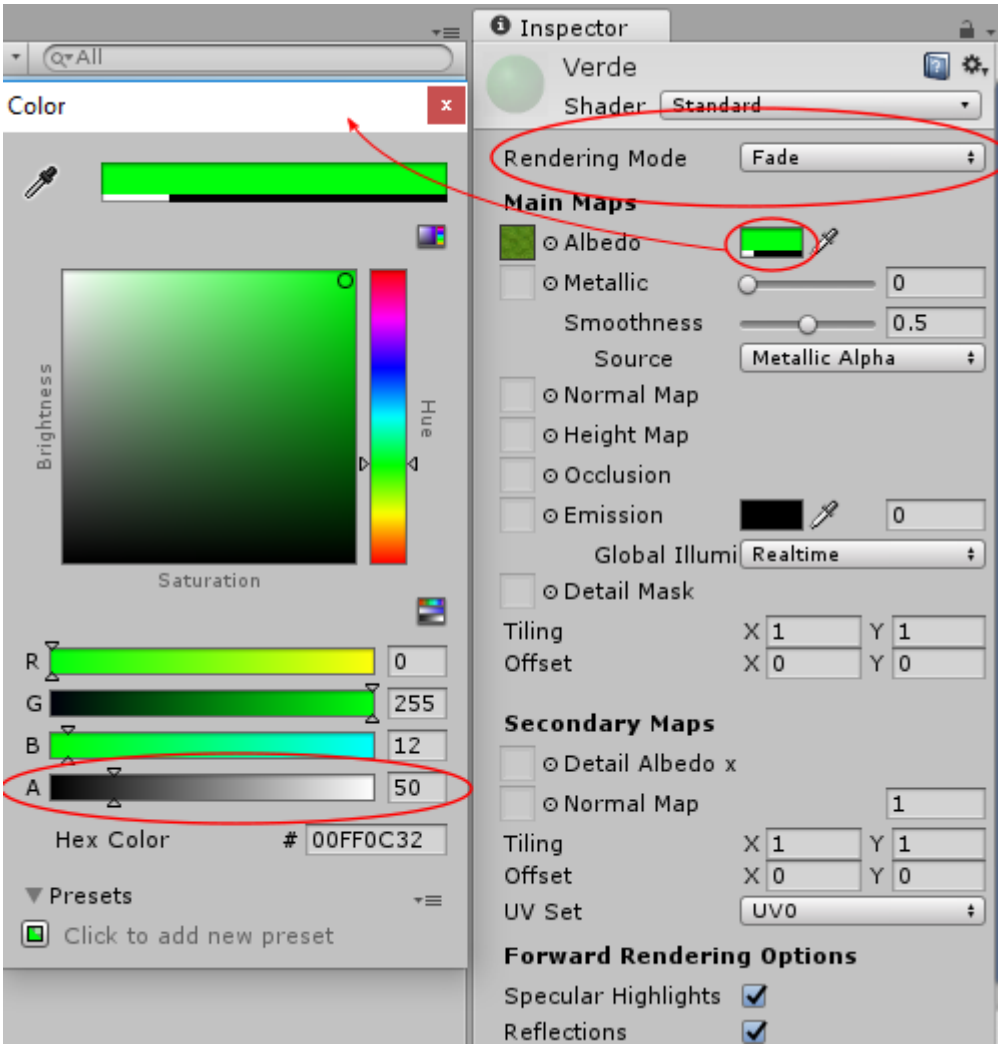
Isso parece uma forma de baixa qualidade de tratarmos a transparência, mas, na verdade, se você puder utilizar esse modo, a grande vantagem é que ele é muito rápido, pois não precisa calcular semitransparências durante o jogo.

Fade

O **Rendering Mode Fade** permite você deixar um objeto com qualquer nível de transparência, de totalmente opaco até totalmente transparente, de modo a poder fazê-lo sumir completamente da cena, inclusive com seus reflexos gerados pela ação da luz na superfície. Esse modo tem uma renderização rápida, porém, ele não é realista quando você deseja representar materiais naturalmente transparentes, como plásticos translúcidos, vidro, etc. Para esses outros materiais, você deve usar o modo **Transparent**.

Modificaremos o Material Verde escolhendo o **Rendering Mode** para **Fade** e, depois, alterando a cor base do Albedo dele ao configurarmos o canal Alpha (A) para 50. Veja a **Figura 9**.

Figura 09 - Material Verde com o Rendering Mode Fade e o canal Alpha da cor base em 50.



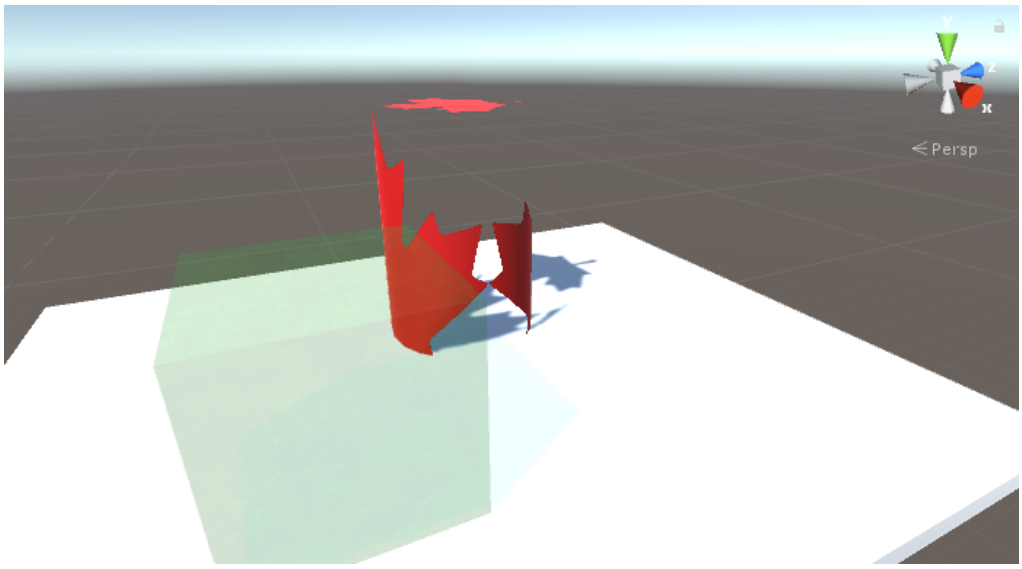
Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Essa mudança fará o cubo no qual esse Material está aplicado ficar translúcido, quase sumindo, já que escolhemos um valor 50 para o Alpha da cor base e o **Rendering Mode Fade**.

Não esqueça de que o valor 50 do Alpha não significa 50%, pois os valores possíveis vão de 0 a 255.

Na **Figura 10** você pode observar o efeito causado no cubo com o Material Verde. Repare que ele fica mais transparente por igual. Isso não é exatamente um efeito realista de um cubo de vidro, pois parece mais com algo que está sumindo gradualmente da cena. Seria um cubo fantasma?

Figura 10 - Cubo com Material configurado com o Rendering Mode Fade e o canal Alpha da cor configurado em 50.



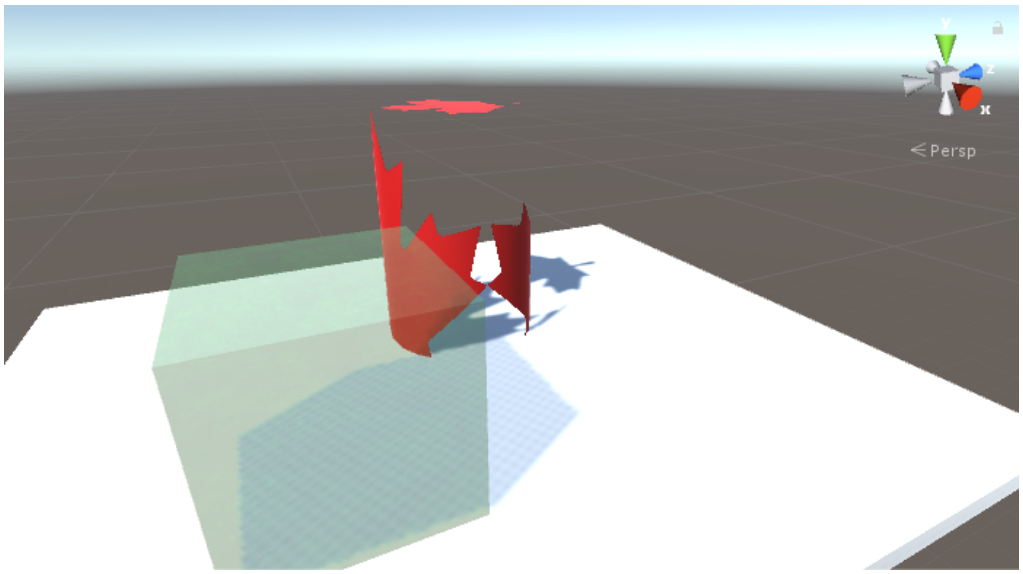
Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Transparent

O **Rendering Mode Transparent** é similar ao **Fade**, entretanto, ao contrário deste, tem um aspecto realista, como de um vidro ou plástico translúcido. Não existe o conceito de melhor ou pior quando se está escolhendo entre **Transparent** ou **Fade**, cada um pode ser utilizado no seu jogo em situações diferentes.

Troque o Rendering Mode do Material Verde para Transparente e mantenha o canal Alpha da cor base em 50 (se já não estiver). Veja o resultado dessas alterações na **Figura 11**.

Figura 11 - Cubo com Material configurado com o Rendering Mode Transparent e o canal Alpha da cor configurado em 50.

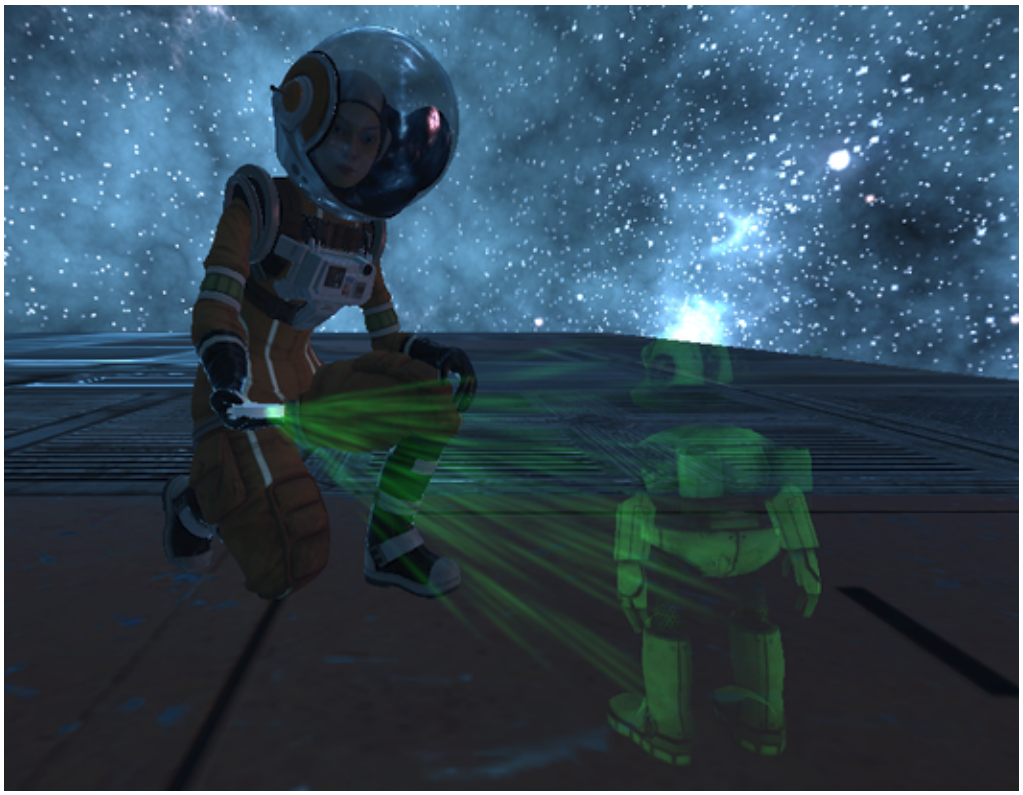


Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Compare as **Figuras 10 e 11** e veja que existe uma diferença de exibição. Na **Figura 11**, embora o Material esteja translúcido, você percebe a existência ainda maior de detalhes nas sombras e superfícies do cubo em relação aos efeitos da luz, tornando-o similar a um cubo de vidro, enquanto no modo **Fade** (**Figura 10**) o cubo é mais simples e semelhante a algo que está sumindo da cena.

Veja, na **Figura 12**, um exemplo de uma cena na qual o vidro do capacete do astronauta tem um Material com o **Rendering Mode Transparent**, enquanto o holograma é **Fade**.

Figura 12 - Imagem de astronauta e holograma, mostrando a aplicação dos modos Transparent e Fade.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://docs.unity3d.com/StandardShaderFadeHologram.png. Acesso em: 03 de abril de 2017.

Albedo

Determina a cor base de um Material, a qual pode ser uma cor sólida ou uma textura (Imagem). Dependendo do Rendering Mode escolhido, os dois valores (Textura e Cor) precisam ser configurados. Vimos, por exemplo, nos **Rendering Mode Fade e Transparent** que você pode escolher uma textura (na aula passada, escolhemos a de grama no Material Verde) e, mesmo assim, precisa, ainda, configurar a cor para determinar seu canal Alpha e escolher o nível de transparência do Material.

Agora, realizaremos uma mudança:

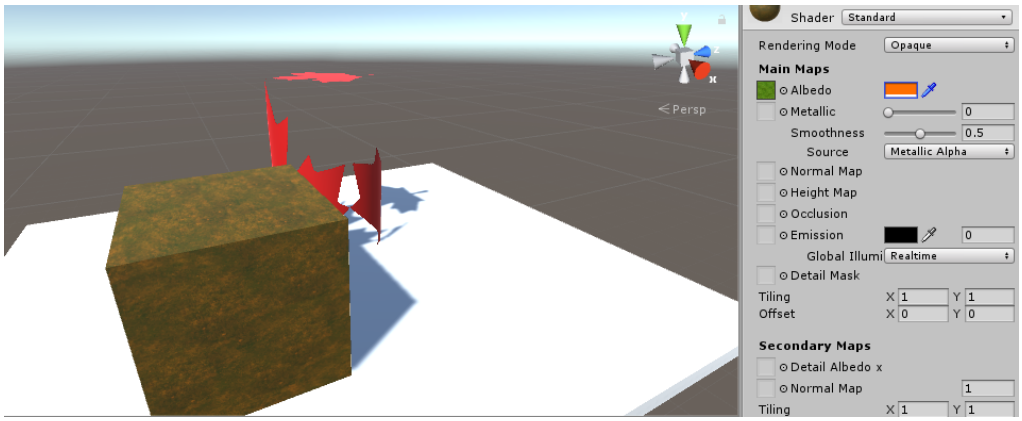
1. Selecione o Material Verde, o qual já deve estar com a cor base verde e com uma textura de grama correspondente a uma imagem importada por você no projeto. Lembre-se de que a imagem da grama já tem naturalmente um tom verde.

2. Troque o **Rendering Mode** para **Opaque** novamente a fim de remover a transparência do Material.

3. Modifique a cor base e volte o canal Alpha (A) para 255. Logo depois, troque a cor e escolha um tom de laranja.

Feche a janela de cor e observe no Scene View que agora seu cubo, o qual tinha esse Material aplicado, tem uma textura de grama laranja, como mostra a **Figura 13**.

Figura 13 - Material "Verde" configurado com uma textura de grama, porém com uma cor base laranja.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

O importante é notar que tanto a textura escolhida no Albedo quanto a cor base afetam o aspecto final do Material. Elas são na verdade "misturadas" entre si, sendo esse processo chamado de **Blending**.

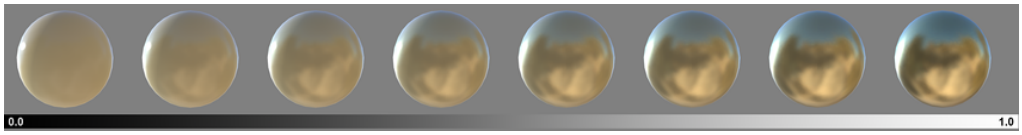
Esse recurso é muito útil, pois você pode ter uma textura base, digamos que em tons de cinza, e pode criar vários Materials com ela, trocando a cor base do Albedo e usando a mesma textura cinza para criar muitos Materials com essa textura em diversas cores.

Metallic

Determina a refletividade da superfície do Material. Não é necessariamente utilizado para objetos metálicos, já que várias superfícies podem ter uma refletividade alta e não ser obrigatoriamente um metal.

Veja na **Figura 14** a variação dos valores do atributo **Metallic** em um Material. Repare que, quanto mais ele reflete as imagens que estão ao seu redor, menos reflete a sua própria cor.

Figura 14 - Variação da propriedade Metallic em um Material.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

O atributo Metallic varia de 0 a 1 e pode ser escolhido arrastando o “slider” ou digitando um novo valor, como exibe a Figura 15. Lembre-se de que, por padrão, ele é 0 para todo Material.

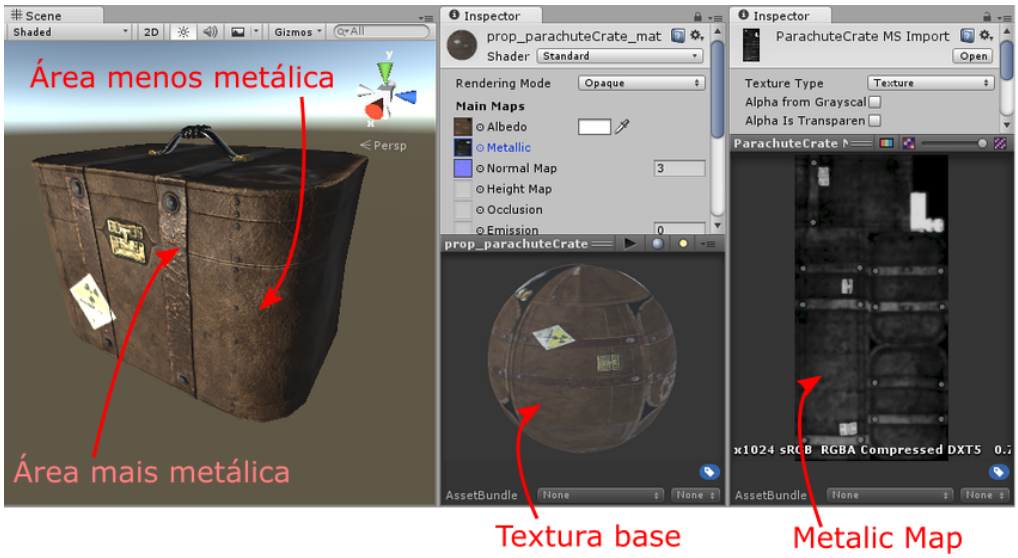
Figura 15 - Atributo Metallic com o valor 0.4 digitado.

Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Além do valor entre 0-1, você pode escolher uma textura especial para determinar quais áreas do Material são mais ou menos Metallicas, fazendo apenas determinadas regiões da textura base receberem esse efeito. Por exemplo, se você está criando um Material para uma mala de viagens, certamente deseja que as partes mais Metallicas da mala tenham um atributo **Metallic** maior do que as partes de couro ou tecido.

Isso é chamado de **Metallic Map** e pode ser escolhido clicando no ícone  ao lado do atributo **Metallic**. Sendo assim, você pode criar uma textura especial para determinar que áreas são essas. Veja na **Figura 16** um exemplo de uma textura aplicada a um objeto com o **Metallic Map** configurado.

Figura 16 - Metallic Map aplicada a um Material.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Sabendo que criar texturas especiais como essa do **Metallic Map** é um trabalho mais adequado ao Modelador 3D e ao designer do jogo, nesta disciplina focaremos mais no uso desses assets do que na sua criação. Portanto, se você gosta do assunto relativo à criação de texturas, procure estudar o assunto mais profundamente.

Atividade 01

1. Crie um novo Material chamado VidroAzul e configure-o para que tenha uma cor base Azul com 50% de transparência.

2. Quais atributos você utilizou para obter o resultado desejado?

Smoothness

Determina o quanto a superfície é rugosa. Um espelho, por exemplo, tem um valor elevado tanto na propriedade **Metallic** quanto em **Smoothness**.

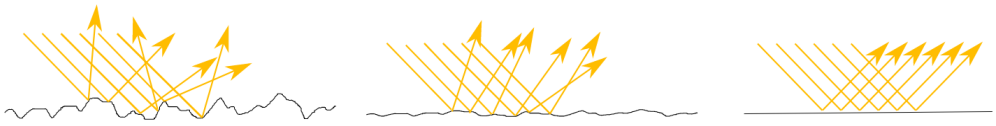
O atributo **Smoothness** pode ser visto na **Figura 17**.

Figura 17 - Atributo Smoothness.

Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Como observaremos, ele pode ser configurado com valores de 0-1. Na **Figura 18** você pode ver uma representação de como os raios de luz são refletidos em superfícies com valores diferentes do Smoothness.

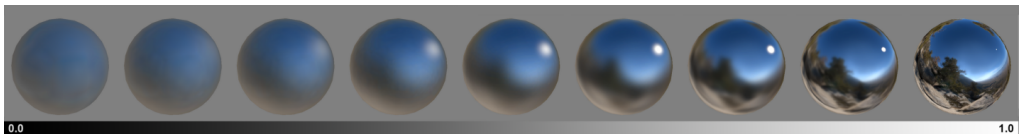
Figura 18 - Reflexos de luz em valores diferentes do Smoothness, de menor para maior.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Na **Figura 19**, você pode ver o efeito do Smoothness, de menor para maior, aplicado a um Material.

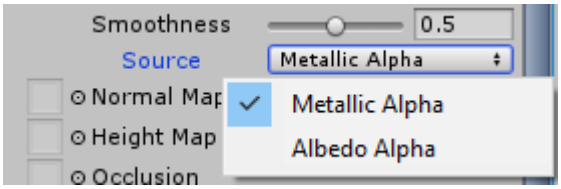
Figura 19 - Efeitos do Smoothness em um Material.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

O atributo Source do **Smoothness** determina em qual textura ele retirará sua máscara, ou seja, que regiões da textura receberão mais ou menos o efeito do **Smoothness**. Pode ser selecionada a opção **Metallic Alpha** ou a opção **Albedo Alpha**. Veja a **Figura 20**.

Figura 20 - Source so Smoothness.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

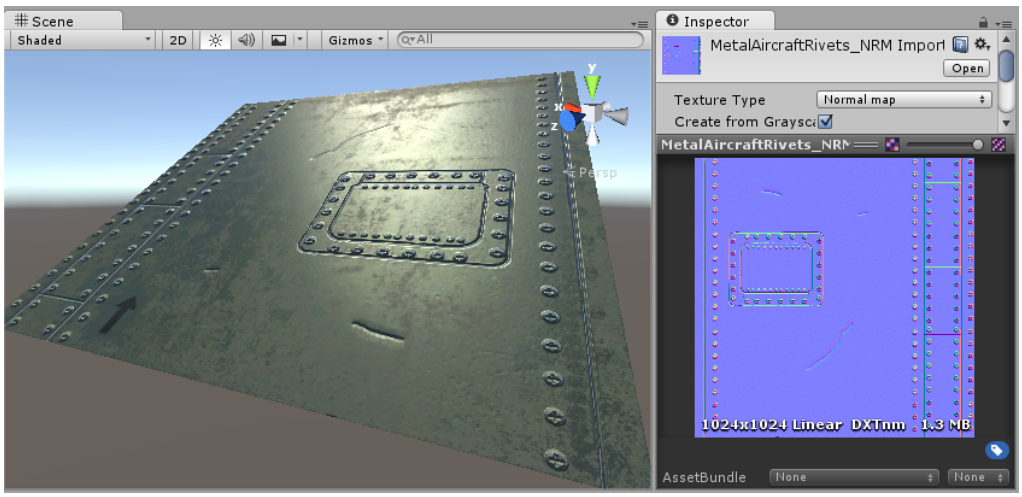
Normal Map

Permite você escolher uma imagem especial que representa os detalhes de superfície de um objeto. É comum, em alguns softwares de edição 3D, esse atributo ter o nome de **Bump Map**.

A textura especial utilizada pelo **Normal Map** normalmente é criada por softwares especializados, cuja avaliação não faz parte do escopo desta aula. Entretanto, é comum no desenvolvimento de um jogo você receber do modelador 3D ou do designer pacotes de texturas já com o **Normal Map**. O **Normal Map** é uma textura (imagem) separada da textura base, possui uma cor variando entre azul e rosa e determina as regiões da textura que terão detalhes mais ou menos elevados.

Na **Figura 21**, você pode ver um exemplo de uma textura de uma placa de metal com parafusos e, à esquerda, o seu **Normal Map** indicando que esses parafusos são mais elevados.

Figura 21 - Textura aplicada a um objeto e, à esquerda, seu Normal Map.

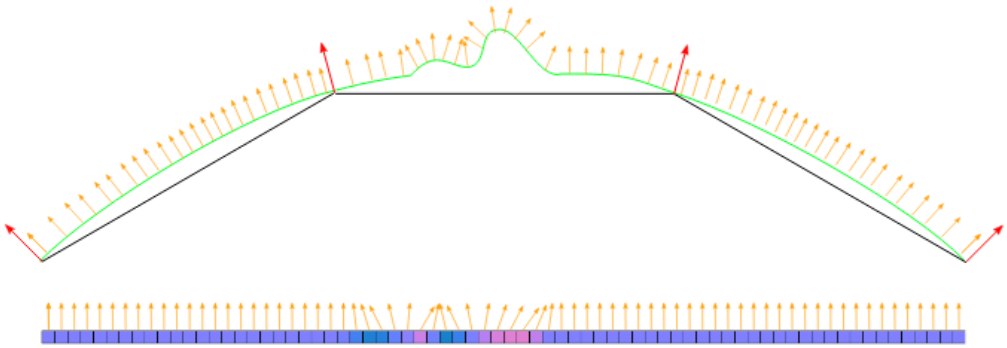


Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Repare que, com a textura na qual o **Normal Map** foi aplicado, você tem uma impressão de os parafusos estarem modelados em 3D, mas na verdade isso é somente uma ilusão, pois, se cada parafuso fosse um objeto 3D, o jogo ficaria muito mais pesado para ser executado.

Na **Figura 22**, você pode ver uma representação da variação dos reflexos de luz de acordo com a variação de cor do **Normal Map** (de azul para rosa). Repare que, nas regiões mais rosa, os raios de luz são refletidos para direções diferentes da normal em que a superfície é menos perpendicular.

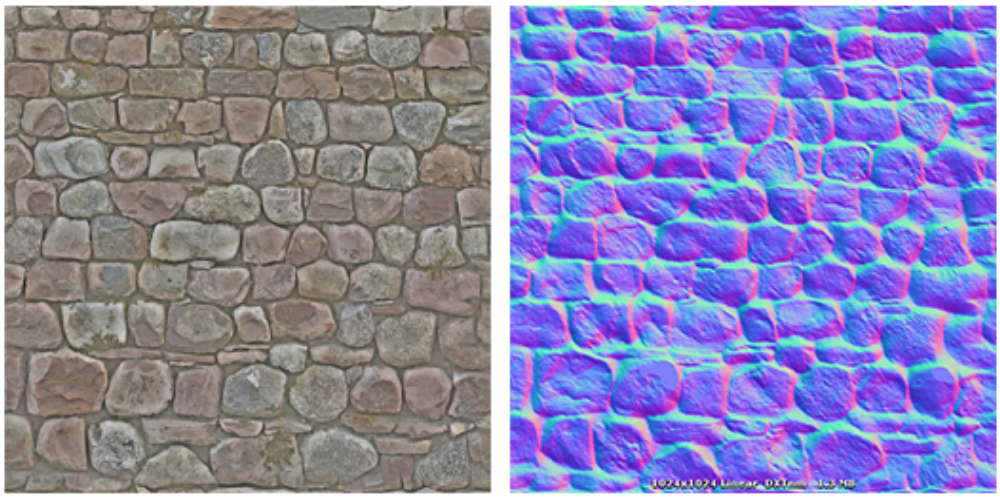
Figura 22 - Variação dos reflexos de luz de acordo com o valor do Normal Map.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Na **Figura 23**, você pode ver a representação das cores referentes a uma textura base de um muro de pedras, ao lado do seu **Normal Map**, o qual representa as saliências da superfície.

Figura 23 - Textura de um muro de pedras com seu Normal Map.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Na **Figura 24**, você pode observar essa textura do muro de pedras aplicada a um objeto sem o **Normal Map** e, logo depois, na **Figura 25**, a mesma textura aplicada com o **Normal Map**. Na **Figura 26**, temos o mesmo exemplo da textura

aplicada com o **Normal Map**, porém em um ambiente noturno com uma tocha próxima ao muro, deixando em maior evidência o efeito realista do **Normal Map** em superfícies.

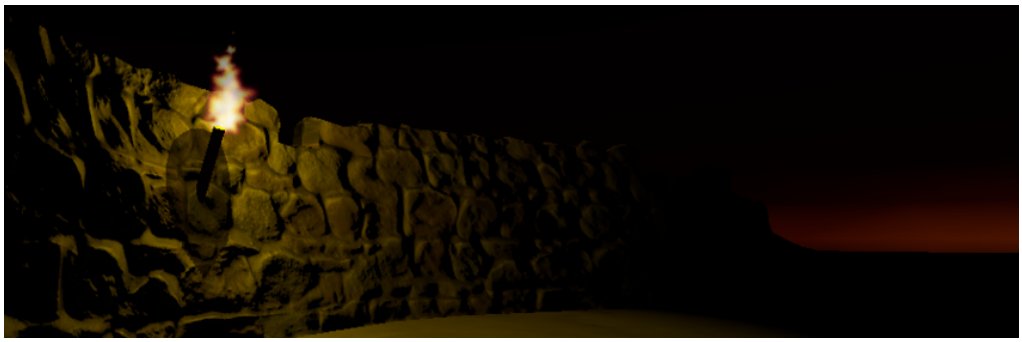
Figura 24 - Textura do muro de pedras aplicada a um objeto sem o Normal Map.

Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Figura 25 - Textura do muro de pedras aplicada a um objeto com o Normal Map.

Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Figura 26 - Textura do muro de pedras aplicada a um objeto com o Normal Map em um ambiente noturno.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

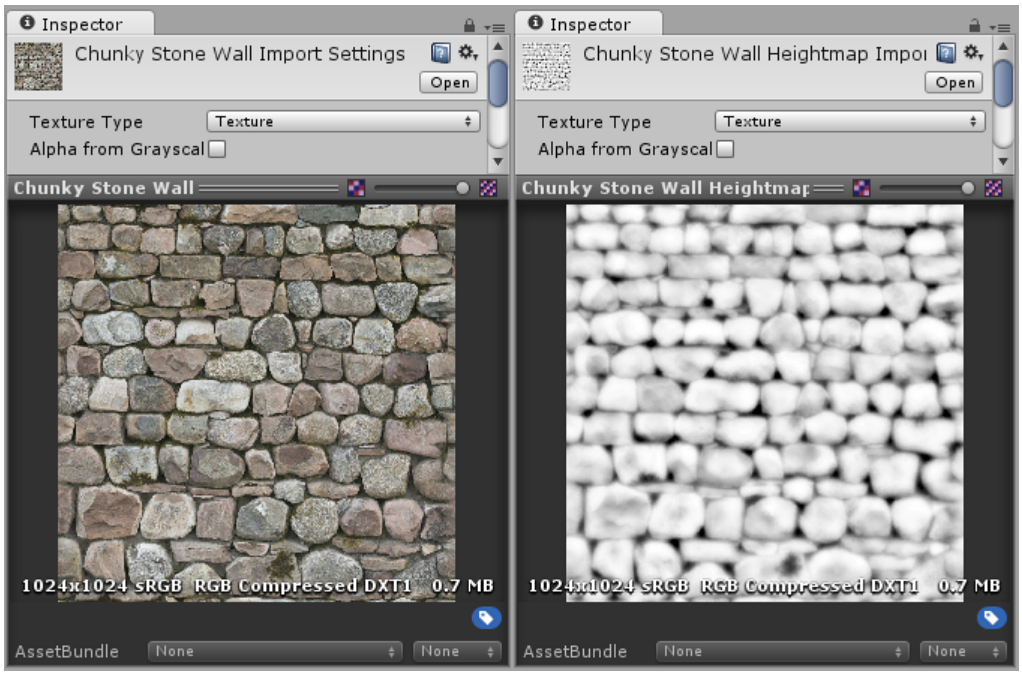
Height Map

O **Height Map** é bastante similar ao **Normal Map**, porém mais utilizado para indicar, através de uma imagem especial, onde estão as maiores protuberâncias de uma textura. Imagine uma textura referente a uma fotografia de areia de praia com uma pegada. Nesse caso, o **Normal Map** seria uma outra imagem detalhando todos os grãos de areia e o **Height Map** seria uma outra imagem indicando que o local da pegada é mais baixo do que o restante da textura. O **Height Map** utiliza um algoritmo mais pesado de cálculo e, por isso, precisa ser realizado separadamente do **Normal Map**, adicionando um grau maior de modificação na superfície. Isso ajuda no realismo da exibição da textura, simulando um aspecto 3D da superfície.

Enquanto o **Normal Map** simplesmente modifica o reflexo dos raios de luz para dar um efeito 3D à sua textura, o **Height Map** dá um passo extra e altera a posição dos pixels da imagem, de maneira que você tenha uma impressão 3D maior, podendo até esconder alguns pixels a depender do ângulo em que esteja vendo a imagem. Esconder? Sim! Imagine-se olhando para um muro de pedras bem salientes, em um ângulo quase paralelo ao muro. Certamente, nesse caso, as rochas mais salientes escondem as menos salientes e que estão mais distantes de você. **Height Maps** são texturas em escala de cinza nas quais as áreas mais brancas são as mais salientes e as mais pretas são as mais profundas.

Na **Figura 27**, você ver um exemplo de uma textura de um muro de pedras e o seu **Height Map** à direita.

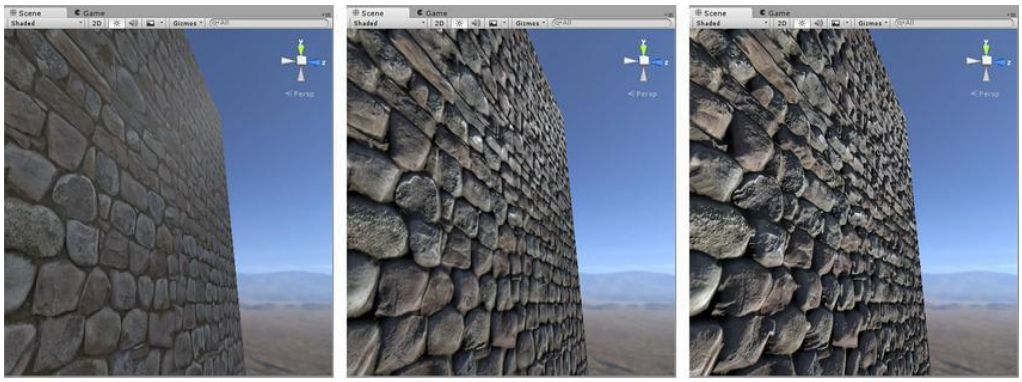
Figura 27 - Textura (à esquerda) com o seu Height Map (à direita).



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Veja na **Figura 28**, essa textura é aplicada a um objeto em uma cena utilizando Materials **sem Normal Map, com Normal Map** e **com Normal Map + Height Map**.

Figura 28 - Exemplo da aplicação da mesma textura em um objeto utilizando três Materials diferentes: sem Normal Map (à esquerda), com Normal Map (centro) e com Normal Map + Height Map (à direita).



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Repare que, na **Figura 28**, o exemplo da direita (**com Normal Map + Height Map**) é muito mais realista e as rochas mais protuberantes chegam a esconder partes das rochas menores que estão por trás.

A textura utilizada no **Height Map** pode, se desejado, também ser utilizada no atributo **Occlusion Map**, o qual veremos a seguir.

Occlusion Map

Occlusion Map é uma textura especial em tons de cinza, similar ao **Height Map**, a qual determina quais regiões da textura do Material receberão mais ou menos luz indireta. Isso cria regiões no seu Material que são naturalmente mais escuras e outras naturalmente mais claras, ajudando o Unity a calcular melhor as sombras e melhorando o realismo da cena.

É normal se utilizar a mesma textura do **Height Map** para o **Occlusion Map**, já que ambas as técnicas utilizam normalmente as mesmas informações de protuberância da superfície da textura.

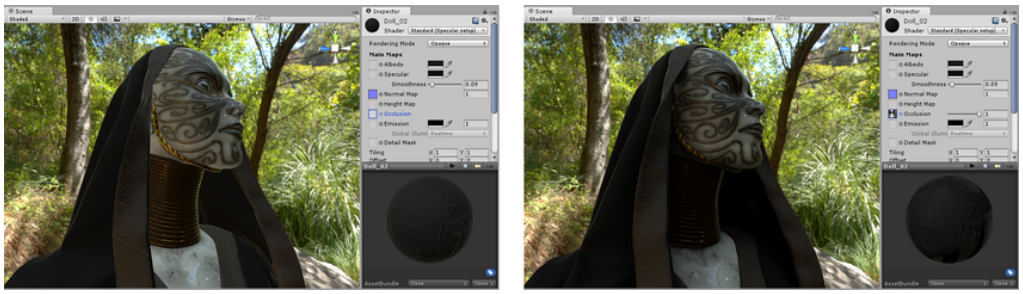
Assim como os outros mapas de textura, o **Occlusion Map** pode ser configurado se escolhendo uma imagem no atributo Occlusion, como visto na **Figura 29**.

Figura 29 - Atributo Occlusion.

Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

Veja na **Figura 30** um exemplo de um personagem mostrado, primeiramente, com um manto sobre a cabeça sem o **Occlusion Map** configurado e, em seguida, com **Occlusion Map** configurado.

Figura 30 - Personagem com textura sem (esquerda) e com (direita) Occlusion Map configurado.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://docs.unity3d.com/StandardShaderOcclusionMap.png. Acesso em: 03 de abril de 2017.

Repare que no exemplo com **Occlusion Map** (à direita), na **Figura 30**, o personagem tem as regiões de sombra do manto mais escurecidas. É uma mudança sutil, mas que contribui muito para dar realismo à cena.

Na **Figura 31**, você pode ver a textura de **Occlusion Map** utilizada no manto do personagem da **Figura 30**.

Figura 31 - Textura de Occlusion Map.

Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://docs.unity3d.com/StandardShaderOcclusionMapTexture.png. Acesso em: 03 de abril de 2017.

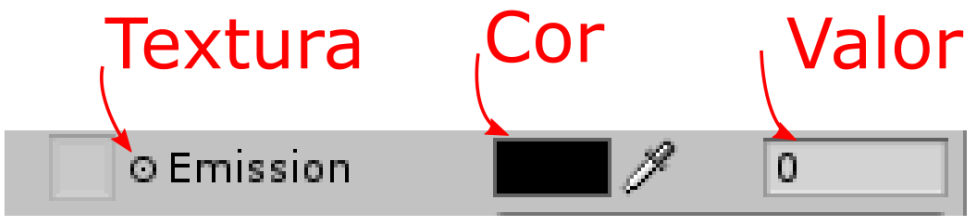
Emission

Controla a cor e a intensidade da luz emitida pelo Material. Com esse parâmetro configurado, seu material é autoiluminado, não precisando de nenhuma outra fonte de luz para que seja visto na cena, além de afetar os outros objetos ao redor.

O atributo **Emission** tem um valor da cor, intensidade e também pode receber uma textura em vez da cor que será utilizada como base para a emissão, ou seja, o Material não só exibe a textura, como também emite raios de luz mais fortes com suas cores. Muito útil se você está, por exemplo, criando uma textura para um monitor de computador que emite luz com mais intensidade do que uma superfície de um objeto não iluminado.

A fim de escolher a cor, você pode simplesmente clicar no ícone de cor do **Emission**; para escolher a textura, pode clicar no ícone de seleção de textura circular; e para escolher o valor basta preencher o desejado no campo de texto. Veja a **Figura 32** com essas propriedades.

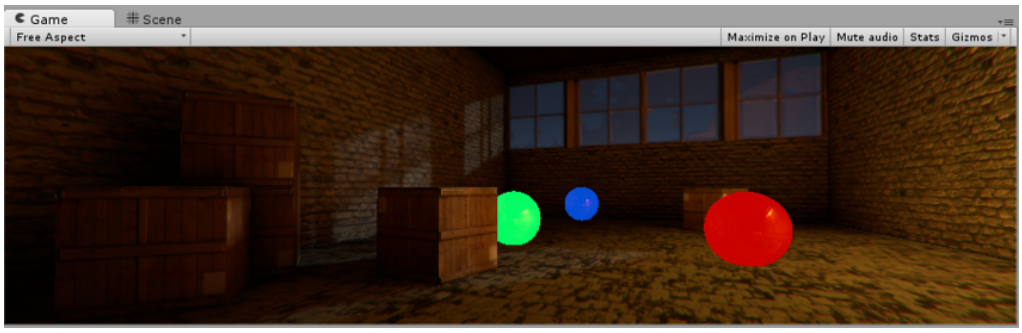
Figura 32 - Atributo Emission.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://unity3d.com/pt/. Acesso em: 03 de abril de 2017.

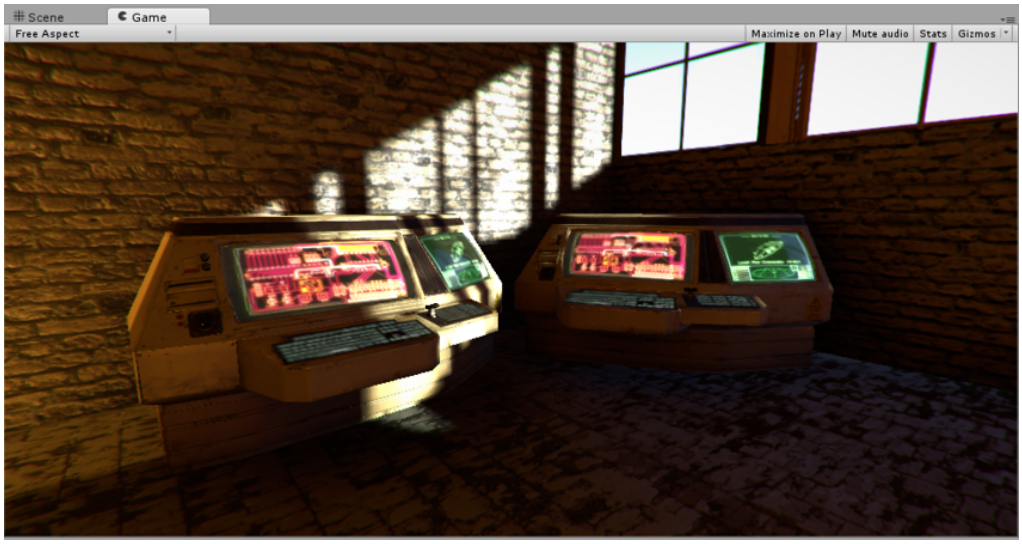
Na **Figura 33**, você pode ver um exemplo de objetos com Materials com **Emission** configurado para cores sólidas e, na **Figura 34**, pode ver um exemplo de um Material aplicado a um objeto com **Emission** de uma textura, sendo nesse caso painéis de um terminal.

Figura 33 - Objetos com Materials com Emission de cores sólidas.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://docs.unity3d.com/StandardShaderEmissiveFlatColour.png. Acesso em: 03 de abril de 2017.

Figura 34 - Objetos com Materials com Emission de texturas



Fonte: Captura de tela do Unity – Game Engine. Disponível em: https://docs.unity3d.com/EmissiveInLightAndShadow.png. Acesso em: 03 de abril de 2017.

O valor do atributo Emission pode ir de 0-1 para uma emissão de iluminação normal de nenhuma (0) à máxima (1), mas também pode ser configurado acima do valor 1, emitindo mais luz do que o valor máximo. Isso parece estranho, mas na verdade é a base de uma tecnologia chamada HDR (High Dynamic Range) presente em algumas TVs e monitores modernos. O HDR possibilita a uma imagem ter mais detalhes sobre iluminação do que normalmente é exibido em um monitor tradicional. Com esse valor acima de 1, os monitores HDR são instruídos a criar um efeito mais forte de emissão de luz naquele local do que o máximo base.

Resumo

Nesta aula, aprendemos alguns parâmetros mais avançados de Materials, discutindo individualmente cada um deles, bem como os seus efeitos visuais nos objetos da cena. Aprendemos, também, o modo como combinar texturas e cores sólidas em Materials, assim como a maneira de alternar entre os diferentes tipos de Rendering Modes disponíveis (Opaque, Cutout, Fade e Transparent).

Leitura Complementar

O assunto de Materials é muito extenso e requer bastante estudo para ser dominado. Naturalmente, nas nossas aulas vimos os conceitos básicos e intermediários, mas cabe a você se aprofundar mais no assunto. Seguem algumas fontes de estudo:

- https://docs.unity3d.com/Manual/Materials.html
- https://docs.unity3d.com/Manual/shader-StandardShader.html
- https://docs.unity3d.com/Manual/Materials.html

Autoavaliação

1. Selecione o Material Verde e aumente o valor do seu Emission para 0.5. Reparou que o seu brilho aumentou, como se estivesse emitindo luz?
2. Agora, aumente o valor do Emission para 1, observe seu efeito e, logo em seguida, aumente para 1.5. Se você não está utilizando um monitor com HDR, não deve ter notado nenhuma mudança no visual dos objetos com esse Material, somente que apareceu o nome “HDR” em cima da cor base do Emission. Se utilizar um monitor com HDR, os locais em que há objetos com esse Material terão uma iluminação bem mais forte. Infelizmente, não se pode observar os efeitos do HDR sem monitores HDR.

Referências

UNITY TECHNOLOGIES. 2016 (C). Unity 3D Online Tutorials [online]. Disponível em: https://unity3d.com/pt/learn/tutorials [Acessado em 16 de novembro de 2016]

UNITY TECHNOLOGIES. 2016 (C). Unity Manual - Prefabs [online]. Disponível em: https://docs.unity3d.com/Manual/Prefabs.html [Acessado em 16 de novembro de 2016].

STOY, C. 2006. Game object component system. In Game Programming Gems 6, Charles River Media, M. Dickheiser, Ed., Páginas 393 a 403.

MARQUES, Paulo; PEDROSO, Hernâni - C# 2.0 . Lisboa: FCA, 2005. ISBN 978-972-722 208-8

UNITY TECHNOLOGIES. 2016 (C). Unity 3D Manual [online]. Disponível em: https://docs.unity3d.com/Manual/index.html [Acessado em 16 de novembro de 2016].