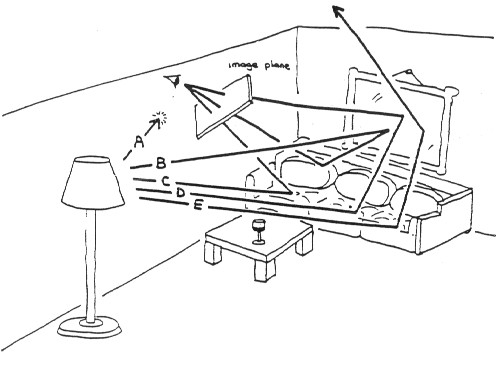
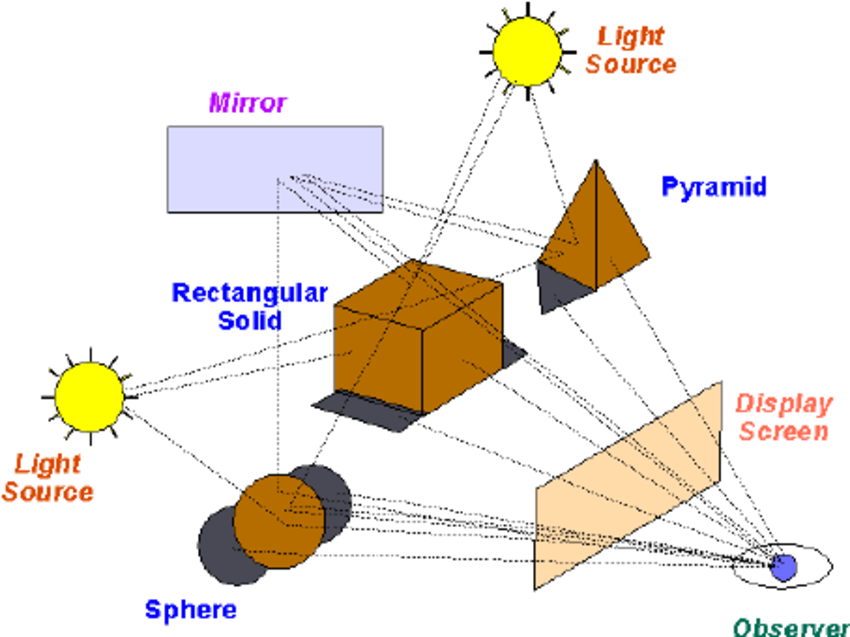
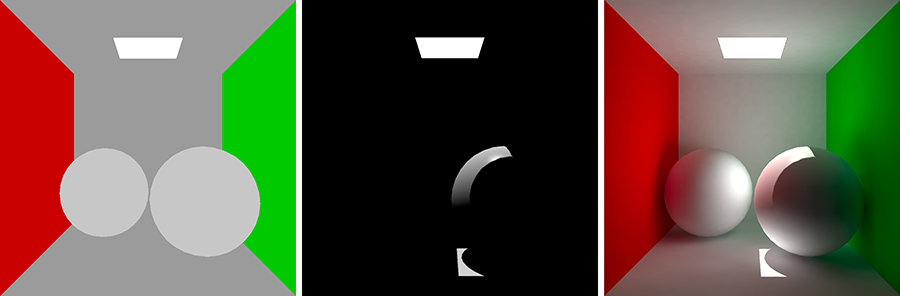
Computação Gráfica para Jogos Digitais

Aula 9: Iluminação

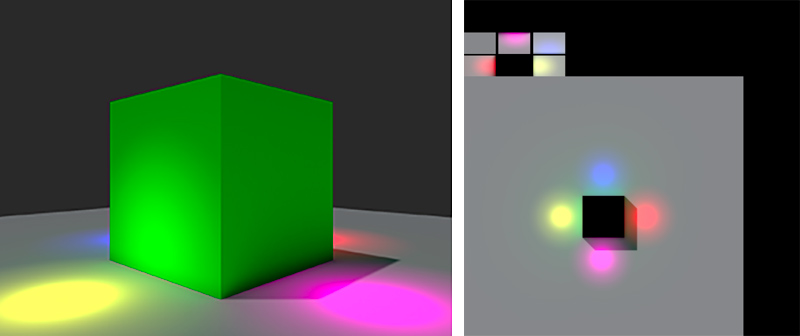
Competências Abordadas:

Para produzir imagens realísticas é necessário simular a aparência de uma superfície sob as diversas condições de iluminação. Ou seja, dada a luz incidida sobre uma superfície, um modelo de iluminação quantifica a quantidade de luz refletida. Os efeitos de iluminação descritos pelos modelos consideram a interação da luz com a superfície das superfícies e utilizam os seguintes fatores:  
 1. Os parâmetros das fontes de luz:  
 ◦ Posições  
 ◦ Espectro Eletromagnético  
 ◦ Formato  
 2. Os parâmetros das superfícies:  
 ◦ Posição  
 ◦ Propriedades de reflexão  
 ◦ Posição das superfícies vizinhas  
 3. Posicionamento da Câmera  
 ◦ Posição/Orientação  
 ◦ Sensibilidade do sensor  


Global Illumination  
A Iluminação Global (GI), também conhecida como iluminação indireta, é um conjunto de algoritmos utilizados em Computação Gráfica 3D que foi construída com o objetivo de criar cenas renderizadas mais realistas. Os algoritmos desse grupo não apenas consideram a luz das fontes de luzes (iluminação direta), mas também consideram os casos subsequentes em que os raios de luz de uma mesma fonte é refletido em outras superfícies, seja essa superfície reflexiva ou não (iluminação indireta).  
  
Teoricamente, reflexões, refrações e sombras são exemplos de iluminação global, já que quando simulando-as, um objeto afeta a renderização de outro. Na prática, apenas a simulação da reflexão difusa (diffuse inter-reflection) é chamada de iluminação global.  
  
Algoritmos  
Imagens renderizadas utilizando iluminação global comumente são mais fotorealistas que imagens renderizadas utilizando apenas iluminação direta. Entretanto, essas imagens são computacionalmente mais caras e consequentemente muito mais lentas de serem geradas. Uma forma comum é computar a iluminação global de uma cena e guardar essa informação com a geometria (radiosidade - radiosity). Essa informação guardada pode então ser utilizada para gerar imagens a partir de diferente pontos de visão sem ter que recalcular toda a iluminação de forma repetida.  
  
São comumente utilizados os algoritmos: Radiosity, ray tracing, beam tracing, cone tracing, path tracing, Metropolis light transport, ambient occlusion, photon mapping, image based lighting, dentre outros. Inclusive é comum que sejam utilizados diversos algoritmos já em alguns casos um algoritmo é rápido, mas não é preciso, enquanto em outros tem uma alta fidelidade, mas ele é lento. Esses algoritmos modelam a reflexão difusa, entretanto, também (exceto o radiosity) modelam a reflexão especular (specular reflection), o que os torna esses algoritmos mais precisos no cálculo da iluminação. Os algoritmos usados para calcular a distribuição da energia da luz entre as superfícies da cena são próximos aos relacionados aos utilizados para simular transferências de calor utilizando métodos elementos finitos.

GI na Unity  
Tradicionalmente na Unity (desde a 5.0 aparentemente) utiliza o modelo de iluminação global. Nas imagens a seguir temos a mesma cena. Na direita temos a cena sem iluminação, no centro apenas a luz direta, na esquerda temos a cena iluminada com a luz indireta. Observe como a cor é "transferida" quando a luz reflete entre as superfícies, obtendo assim um resultado muito mais realista.  
  
  
A Iluminação na Unity pode ser considerada tanto "realtime" quanto "précomputada" e ambas as técnicas podem ser utilizadas de forma combinadas.

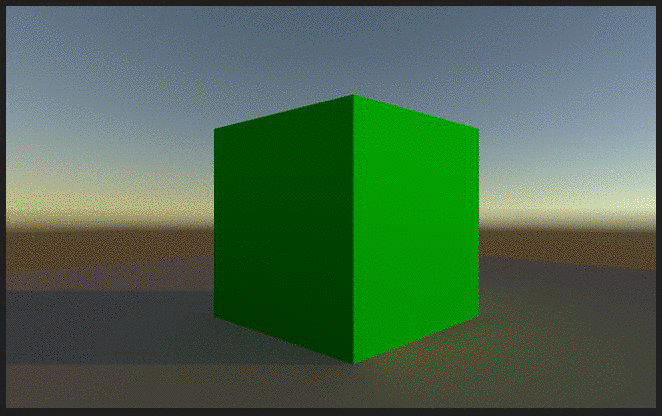
"O Bake"  
Para reduzir o custo de execução da iluminação global a Unity faz o que chamamos de "bake de iluminação" também chamada de "Baked Lightmaps". Grande parte dos objetos em uma cena comumente são estáticos como paredes, tetos, chão, móveis dentre outros, para esses objetos é possível calcular antecipadamente a luz indireta e ela continuará correta durante o tempo de execução. Outra grande vantagem é que utiliza um poder computacional maior para gerar sombras macias em áreas iluminadas.



Quando "assando" o mapa de iluminação (lightmap), os efeitos das luzes estáticas nos objetos da cena são calculados e o resultados são escritos nas texturas que são então sobrepostas para criar a simulação da iluminação. Esses mapas de iluminação podem incluir tanto luzes diretas que incidem diretamente sobre a superfície, assim como a luz indireta que reflete de outros objetos presentes na cena.  
  
A luzes em tempo real podem ser sobrepostas e serem utilizadas de forma aditiva sobre a textura de iluminação, mas não podem de forma interativa mudar a textura de iluminação.  
Sequencia de Passos

1. Create Geometry  
 2. Atlasing  
 3. Create Baked Systems  
 4. Baked Resources  
 5. Bake AO  
 6. Export Baked Texture  
 7. Bake Visibility  
 8. Bake Direct  
 9. Ambient and Emissive  
 10. Create Bake Systems  
 11. Bake Runtime  
 12. Upsampling Visibility  
 13. Bake Indirect  
 14. Final Gather  
 15. Bake ProbesSet  
 16. Compositing

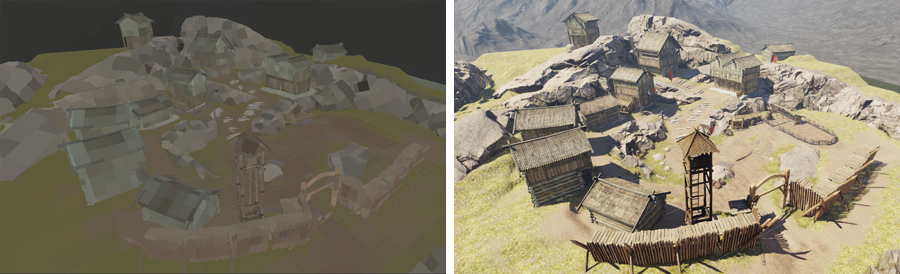
"Precomputed Realtime GI"

No bake são utilizados os tradicionais mapas de iluminação estáticos, é comum a utilização da técnica de Precomputed Realtime GI. Essa técnica permite que a cena tenha a iluminação atualizada de forma interativa. Através dessa técnica é possível simular luzes que refletem, assim que a iluminação muda.  


Essa técnica também requer uma pré-computação e também só funciona com objetos estáticos, entretanto, não apenas pré-computa como a luz reflete na cena, mas onde irá refletir caso exista uma superfície seja iluminada. A Unity salva essa informação sobre qual o caminho a luz pode propagar para uso futuro. Dessa forma a iluminação final feita em tempo real utiliza esses caminhos para reduzir o custo computacional da iluminação em tempo real.

*“Additionally, Unity 5.0 adds support for a new technique called Precomputed Realtime GI. It still requires a precomputation phase similar to the bake mentioned above, and it is still limited to static objects. However it doesn’t just precompute how light bounces in the scene at the time it is built, but rather it precomputes all possible light bounces and encodes this information for use at runtime. So essentially for all static objects it answers the question “if any light hits this surface, where does it bounce to?”*

Isso significa que o número de luzes presentes na cena, o posicionamento das mesmas, as direções assim como outras propriedades podem ser alteradas sem problemas para ser atualizada. Similarmente, também é possível trocar as propriedades do um material de um objeto.  
  
Para acelerar a pré-computação a Unity não trabalha com mapas de texturas quando está fazendo precomputed realtime GI, mas ela cria uma ambiente de baixa resolução que é uma aproximação da geometria estática do mundo, chamada de clusters. Na imagem a seguir temos à esquerda uma cena construída com o 'Albedo' dos texels gerados pela Unity Precomputed Realtime GI. Por padrão, o texel nessa cena é aproximadamente do mesmo tamanho de um cluster. Na cena da direita, temos como a cena aparece em jogo quando a iluminação foi calculada e os resultados convertidos em um mapa de textura que foi aplicado

  
  
Tradicionalmente quando é calculada a iluminação global de uma cena é utilizado um algoritmo similar ao ray tracing. Por esse motivo é muito custoso e demanda muitos recursos para ser atualizado em tempo real. Entretanto a Unity utiliza ray tracing para calcular as relações entre as superfícies do cluster durante o estágio de "Light Transport".

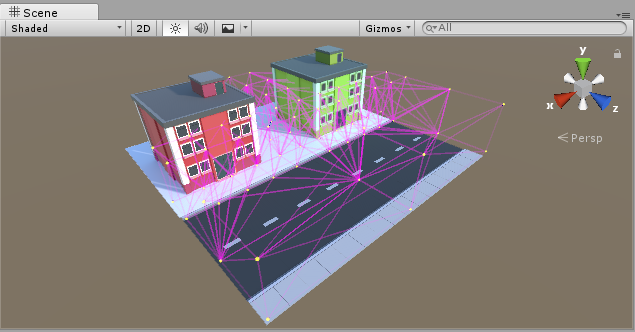
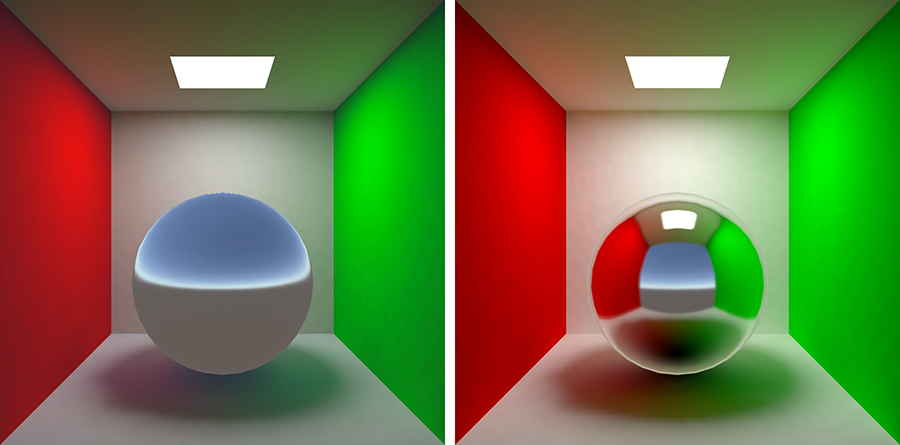
“By simplifying the world into a network of relationships, we remove the need for expensive ray tracing during the performance-critical gameplay processes.  
We have effectively created a simplified mathematical model of the world which can be fed different input during gameplay. This means we can make modifications to lights, or surface colors within the scene and quickly see the effects of GI in scene lighting update at interactive framerates. The resulting output from our lighting model can then be turned into lightmap textures for rendering on the GPU, blended with other lighting and surface maps, processed for effects and finally output to the screen.”

A Precomputed Realtime GI gera também sombras macias,entretanto ela será mais granularizada que as sombras geradas por Bake, exceto para cenas muito pequenas. Observe que enquanto Precomputed RealtimeGI faz a iluminação final em tempo de execução, ou seja, é feita de forma iterativamente ao longos de vários frames, portanto, se uma mudança muito grande for feita na iluminação, irá demorar mais frames para ter o efeito. Normalmente, isso não é um problema, exceto se você estiver trabalhando em ambientes com poucos recursos.

Sequencia de passos

1. Create Geometry  
 2. Layout Systems  
 3. Create Systems  
 4. Create Atlas  
 5. Clustering  
 6. Visibility  
 7. Light Transport  
 8. Tetrahedralize Probes  
 9. Create Probe Set

Limitações da GI & Benefícios & Custos na Unity  
Tanto a iluminação "bake" e a iluminação "précomputada em tempo real" funcionam apenas para objetos marcados como estáticos, portanto objetos que se movem não podem refletir luz em objetos que não se movem e vice e versa. Entretanto, é possível utilizar Light Probes para calcular a reflexão em objetos que se movem de objetos estáticos. Light Probes são posições em uma cena em que a luz é medida (probed) durante o bake/precomputação e durante a execução a luz indireta que acerta objetos não estáticos é aproximada. Por exemplo, se uma bola vermelha rola ao lado de uma parede branca, não irá "sangrar" (bleed) sua cor na parade, mas uma bola branca próxima a uma parede vermelha irá receber essa coloração avermelhada com o uso de light probes.  
  
É possível utilizar também tanto a iluminação Bake quanto a Pré computada juntas, entretanto, o custo será a soma dos dois. Não apenas é necessário adicionar os mapas de iluminação na memória, quanto o custo de decodificar em ambos.  
  
Tanto a Baked GI quanto a Precomputed Realtime GI, guarda informações sobre sua cena enquanto no 'GI cache' e irá tentar utilizar essas informações sempre que possível para reduzir o tempo de processamento. O número e a natureza das mudanças que você fizer na sua cena irá determinar o quanto de informações podem ser reusadas. Esse arquivo é guardado fora do projeto da Unity e pode ser limpo em Preference > GI Cache > Clear Cache.   
  
Isso pode levar a situações em que talvez seja preferível um tipo de iluminação. É comum que em jogos mobile exista pouca memória gráfica disponível e o custo de processamento é muito limitado, portanto é muito mais recomendado utilizar uma iluminação do tipo Bake. Já em computadores e consoles é comum escolher a pré-computada, ou ainda combinar ambos os sistemas.  
  
Destaca-se que o LOD (Level Of Detail) é levado em consideração quando a Unity gera os mapas de iluminação. A luz direta é computada utilizando as superfícies de todos os objetos em todos os leveis de detalhamento. Baixos leveis de detalhamento utilizam light probes para capturar a luz indireta. O iluminação resultante é colocada dentro da textura de iluminação. Portanto, é importante colocar light probes próximos em volta dos LODs para capturar a luz indireta e o objeto não irá usar lightprobes em tempo de execução se você utilizar uma iluminação completamente baked.  
  
Lightprobes & Reflection Probes  
Light Probes permitem uma forma de capturar e usar informações de luz que são passadas em espaços vazios na cena. Similar aos lightmaps, os light probes guardam informações sobre a luz na cena, a diferença é que enquanto lightmaps guardam informações sobre a luz que acerta as superfícies na cena, os light probes guardam informações sobre as áreas vazias. O principal uso de light probes é promover uma iluminação de qualidade (incluindo luzes refletidas) para objetos que se movem na cena. O segundo uso é guardar informações de luz para cenários estáticos para o LOD.  
  
A resolução da informação sendo obtida e a quantidade de computação feita enquanto o jogo está rodando é totalmente definida pela quantidade de light probes presentes na cena. Portanto, quanto menos light probes, melhor o desempenho. Entretanto, em áreas em que a iluminação é bastante complexa talvez seja interessante posicionar muitos.  
  
Na documentação oficial é apresentado o seguinte exemplo apresentado na imagem abaixo. Na região mais densa da cidade são colocados mais light probes do que nas regiões mais simples.

  
  
Reflection probes são utilizados para calcular a reflexão em superfícies. O ambiente dado um ponto na cena pode ser representado por 1 cubemap. Conceitualmente é como uma caixa com imagens planas de cada uma das faces internas do cubo. Para um objeto ter reflexões é necessário que o shader tenha acesso às imagens representando o cubemap.   
Cada ponto da superfície do objeto pode “ver” uma pequena área do cubemap na direção em que a superfície está voltada (ou seja, a direção do vetor normal da superfície). O shader usa a cor do cubemap nesse ponto para calcular a cor que a superfície do objeto deve ter; um material de espelho pode refletir a cor exatamente enquanto um carro brilhante pode desvanecer e tingir um pouco.

Passos no Bake & Precomputed  
 1. Ambient Probes  
 2. Baked/Realtime Ref.Probes