Portais Renderizáveis em 3D

Alunos:
Omar Mahmoud Abou Ajoue
Otávio Moura do Nascimento

Orientador:
Marcel P. Jackowski

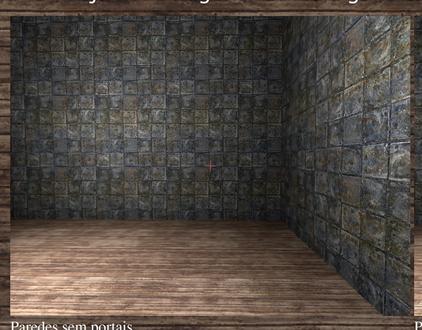
Introdução

Além da convencional movimentação em mundos 3D, em que um personagem é capaz de andar, correr e pular, inspirados em um jogo chamado Portal(R), criamos uma implementação de portais renderizáveis.

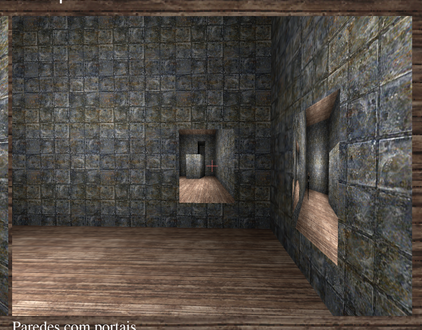
Imagine um mundo 3D em que tarefas simples podem ser importantes, como por exemplo, encontrar uma saída de uma sala. Porém, pode ocorrer que não haja um caminho direto entre o jogador e a saída, como por exemplo, uma barreira de fogo. Como é impossível seguir andando, o jogador pode utilizar portais, que lhe permitem entrar por um ponto e chegar a outro. Isto cria um novo paradigma de desafios que podem ser adicionados ao mundo, permitindo criar diversos quebra-cabeças.

O funcionamento ocorre de forma bastante simples: os botões do mouse, quando disparados, criam portais nas paredes do mundo 3D: um portal para o botão direito e outro para o botão esquerdo. Estes portais estão interligados, de forma a criar um sistema de entrada-saída. Ao se aproximar do portal, o jogador é levado à localização do outro portal.

Veja nas imagens abaixo alguns exemplos:



Paredes sem portais.



Paredes com portais.

Visão do personagem ao se aproximar da parede à direita.

Visão do personagem ao se aproximar da parede à frente.

Objetivos

Para dar realidade aos portais que criamos, é necessário que alguns passos sejam tomados:

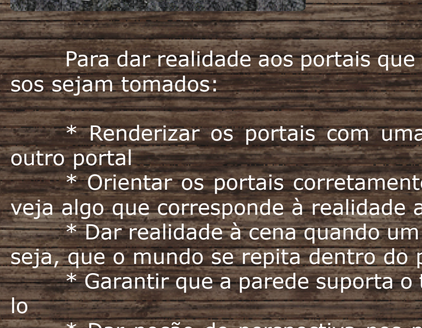
* Renderizar os portais com uma visão do mundo com referência no outro portal
* Orientar os portais corretamente no mundo, para que o personagem veja algo que corresponde à realidade através do portal
* Dar realidade à cena quando um portal consegue visualizar o outro, ou seja, que o mundo se repita dentro do portal
* Garantir que a parede suporta o tamanho do portal antes de posicioná-lo
* Dar noção de perspectiva nos portais: quando o jogador se move, o portal deve representar a realidade do outro lado
* Travessia de um lado a outro

Os passos acima foram todos resolvidos usando diversas técnicas vetoriais, com conceitos de álgebra linear. Descreveremos cada um dos passos a seguir.

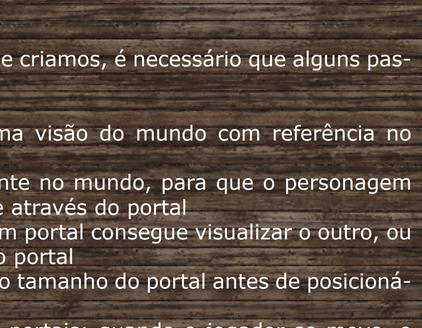
Soluções

1. Renderização

Para a criação de portais, foi utilizada uma técnica chamada Render to Texture (RTT), que possui suporte nos drivers de vídeo mais recentes e a partir da versão 1.3 do OpenGL, desde que sejam utilizadas extensões. Esta técnica nos permite dar vida aos portais, de forma a permitir que o "outro lado" do mundo seja visível para o jogador, como se fosse uma porta para o "outro lado".



Portal antes da aplicação da textura renderizada.



Portal depois da aplicação da textura renderizada.

A técnica de RTT consiste numa manipulação de buffers de vídeo, gerando sua saída para uma textura, ao invés de levá-la à saída convencional: a tela. Sem a existência desta ferramenta, seria muito caro criar texturas "on the fly".

Convencionalmente, texturas são obtidas de imagens ou modelos pré-definidos durante um processo de carga da aplicação e apenas utilizadas em superfícies em tempo de execução. Para dar verossimilhança aos portais, seria necessário criar texturas em tempo de execução, que levariam em conta onde os portais estão localizados.

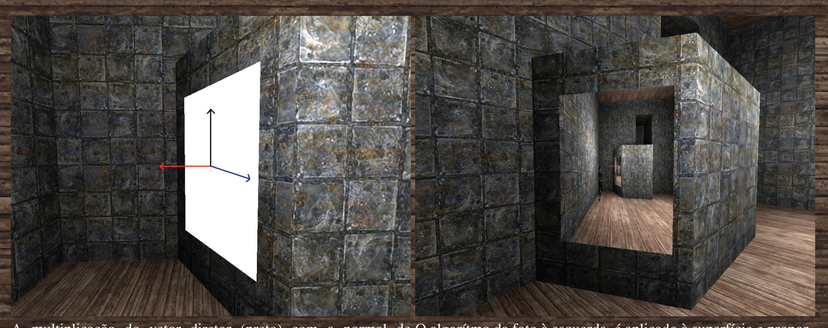
É possível obter parte da saída do frame buffer; poderíamos manipulá-la para direcionar à textura. Mas este processo é muito lento, pois o processo de cópia é desnecessário. Seria possível reduzir o tamanho da área de renderização, facilitando o processo, o que poderia tornar o processo mais rápido, mas ainda não escaparíamos da indesejável cópia.

Utilizando extensões de OpenGL, é possível fazer com que a máquina gráfica renderize diretamente para o alvo desejado: um buffer que será utilizado como textura. O processo ocorre da seguinte forma:

* Criar uma textura de renderização (chamados pixel buffer ou pbuffer), que não é passado à tela final
* Informar à máquina gráfica que o alvo de renderização será o pbuffer criado
* Renderizar a imagem
* Voltar o alvo de renderização à janela
* Ligar o pbuffer ao objeto de textura que utilizaremos
* Utilizar a textura como qualquer outra
* Liberar o pbuffer da textura

2.Orientação dos portais nas superfícies

Não é suficiente apenas conseguir renderizar os portais. É necessário garantir que sua orientação esteja correta, para qualquer parede que utilizamos. Este ajuste precisa ser feito toda vez que um portal é criado, utilizando como base a direção que o personagem está olhando. Uma linha deve ser traçada, a partir do personagem, para encontrar o ponto de colisão com a parede mais próxima à sua frente e utilizá-la como base. Nomearemos esta linha de L, e a normal da parede à frente será chamada de N.



A multiplicação do vetor diretor (preto) com a normal de qualquer parede (vermelho), sempre resultará em um vetor "horizontal" (azul) que pode ser usado para desenharmos o portal.

O algorítmo da foto à esquerda, é aplicado à superfície e proporciona um posicionamento confiável.

A forma que utilizamos para orientar os portais funciona bem para qualquer parede que não seja paralela ao chão (eixo XZ), mas não é difícil fazer um método genérico que permite orientar os portais corretamente para qualquer parede. O método utilizado para o posicionamento dos portais é o seguinte:

Utilizamos um vetor diretor D, com coordenadas (0, 1, 0), ou seja, um vetor que aponta para cima (lembrando que na máquina 3D, o eixo X representa a largura, o eixo Y representa a altura e o eixo Z representa a profundidade).

O produto vetorial deste vetor D com o vetor N (normal da parede) nos gera um vetor que está no plano horizontal no mundo do personagem. Fazendo um novo produto vetorial, deste vetor horizontal com a normal da parede, obtemos um vetor que vai estar sempre "para cima" com relação à parede. Este algoritmo, bastante simples, funciona muito bem para quaisquer paredes que não sejam chão, teto ou paralelas.

A razão deste acontecimento é que, não importa qual vetor diretor D seja utilizado, os portais serão sempre orientados com a mesma referência de baixo e cima. É fácil visualizar o acontecimento imaginando a seguinte cena: escolha um vetor diretor D com coordenadas (0, 0, -1), ou seja, um vetor que está apontando, por padrão, para o "fundo" do mundo. Em qualquer ponto do chão ou teto, a normal será sempre (0, +/-1, 0). Este produto vetorial resulta sempre no mesmo vetor: (+/-1, 0, 0), ou seja, todos os portais terão como orientação horizontal o eixo X e como orientação vertical o eixo Z.

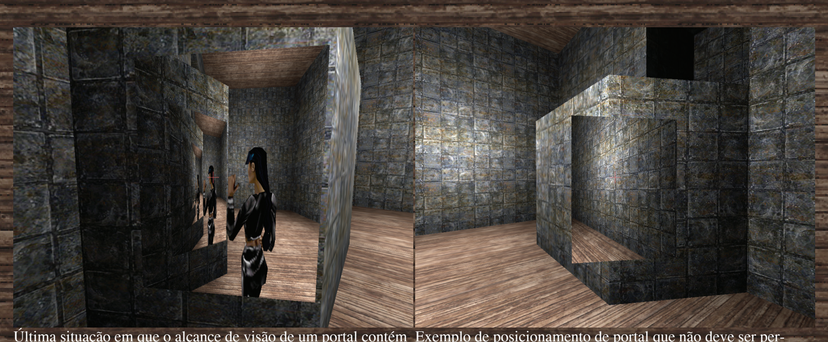
Para resolver o problema do posicionamento dos portais em paredes paralelas ao eixo XZ, são utilizadas projeções de vetores para obtenção dos vetores horizontal e vertical da seguinte forma:

Se traçarmos uma linha que vai do observador até o ponto de intersecção com o chão ou teto, temos um vetor, que chamaremos de O (observador). Projetando este vetor na normal, conseguimos obter sua componente na vertical do mundo. Subtraindo-a do vetor O, seria como se "deitássemos" este vetor no chão ou teto. Utilizando-o como diretor vertical, e utilizando um produto vetorial com a normal, é possível obter um diretor horizontal.

3. Renderização no caso em que um portal é capaz de visualizar o outro

Com as técnicas até agora descritas, é possível criar um portal, renderizá-lo com o mundo visto através do outro portal e posicioná-lo com orientação correta. Porém, é possível que ocorra uma situação em que um portal consegue enxergar o outro, tornando necessário que o mundo seja redesenhado dentro do mundo visto pelo portal.

Uma vez feito o RTT de um portal, é como se tivéssemos tirado uma foto do mundo, com a visão do "outro lado". Se aplicarmos esta textura à parede e realizarmos o processo novamente, teremos uma "foto da foto", ou seja, conseguimos criar uma profundidade necessária para dar vida aos portais.



Última situação em que o alcance de visão de um portal contém o outro e vice-versa.

Exemplo de posicionamento de portal que não deve ser permitido.

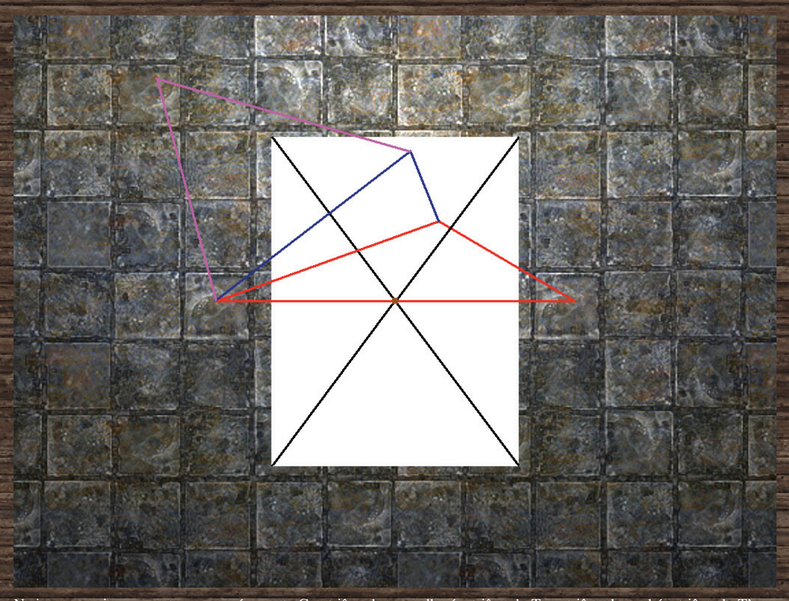
4. Alocação do portal na parede

Durante o processo de criação de um portal, precisamos garantir que há espaço para alocação do mesmo. Devemos então verificar se toda a extensão do portal será devidamente posicionada na parede.

Para isto, poderíamos simplesmente criar mundos 3D em que as paredes são construídas de forma a sempre suportarem os portais. Como não haveriam paredes tortas ou complicadas, todos os portais caberiam facilmente nelas. Ainda com estas limitações, conseguiríamos criar fases interessantes para nossos jogos.

Entretanto, optamos por criar um algoritmo que decide se o portal cabe em uma determinada parede, que funciona em qualquer mundo 3D. Assim, temos a liberdade de usar qualquer cena para utilizarmos nossos portais. Este algoritmo se baseia no fato de que a máquina gráfica divide todas as paredes da cena 3D em triângulos, para que a placa gráfica possa desenhá-las. O processo executado por este algoritmo é descrito a seguir:

* Utilizamos a linha L descrita anteriormente, que encontra o ponto de colisão com a parede mais próxima à frente do jogador e nomeamos este ponto de C.
* Obtemos também o triângulo da parede onde este ponto está contido. Chamaremos este triângulo de T.
* O ponto C será o centro do retângulo que forma o portal.
* A partir do ponto C, criamos quatro linhas, cada uma ligando o ponto C aos pontos onde ficarão os vértices do retângulo.
* Para cada linha, verificamos se a mesma está contida em T.
* Caso uma linha esteja contida em T, isto significa que o portal cabe na parede na direção daquela linha. Caso contrário, achamos o próximo triângulo T' da parede na direção da linha.
* Caso não haja nenhum, ou T' não esteja no mesmo plano de T, o algoritmo termina e descobrimos que o portal não cabe naquela parede.
* Caso contrário, continuamos o algoritmo a partir do novo triângulo.
* Decidimos que o portal cabe na parede quando as 4 linhas tiverem sido exploradas até o final.



Na imagem acima, o ponto marrom é o ponto C, o triângulo vermelho é o triângulo T e o triângulo azul é o triângulo T' obtidos na primeira iteração, quando o algoritmo analiza a linha preta que liga o ponto C ao vértice superior esquerdo do retângulo que forma o portal. Como T' está no mesmo plano que T e a linha ainda não terminou, o algoritmo passa para a próxima iteração, obtendo o próximo triângulo (rosa) da parede na direção da linha. Este novo triângulo está no mesmo plano que o triângulo anterior e, como a linha acaba sem encontrar mais triângulos, o algoritmo decide que o portal cabe nesta parede, na direção desta linha.

5. Perspectiva

Chegamos a um ponto em que conseguimos garantir que o portal está devidamente posicionado, preenchido, com profundidade, mas ainda falta algo: uma noção de perspectiva.

É importante que, quando o jogador se move, o portal de fato demonstre o que acontece do outro lado, para dar mais realidade à cena, ou seja, de acordo com a posição do personagem, um ajuste à câmera que representa o outro lado deve ser feito. O portal deve funcionar como se o jogador estivesse olhando através de uma janela. Quando uma pessoa olha para uma janela e se move para a esquerda, ela consegue enxergar coisas que estavam mais para a direita, no mundo exterior. Queremos reproduzir isto nos portais.

Novamente, utilizamos projeções de vetores para resolver o problema. A solução é bastante simples:

* Obtemos a norma de V, que é o vetor que vai do personagem até o central do portal
* Normalizamos os seguintes vetores: o vetor V, o vetor DH que é o diretor horizontal do portal e o vetor DV que é o diretor vertical do portal
* Calculamos a projeção do vetor V em DH e DV. Desta forma, temos as componentes vertical e horizontal (com relação ao portal) do vetor V. Chamaremos estas projeções de pDH e pDV.
* Subtraimos de N, que é a normal da parede onde o vetor se encontra (também normalizada) os valores pDH e pDV. Assim sendo, temos agora as 3 componentes que formam o vetor V, todas perpendiculares
* Realizamos a mudança de forma correspondente na câmera que está do outro lado, utilizando a norma de pDH, pDV e V - pDH - pDV

O último passo da lista acima é feito da seguinte forma:

Como pDH e pDV são obtidos como múltiplos de DH e DV respectivamente, então, sua norma representa diretamente o quanto "para direita" ou "para cima" o personagem está do portal.

Se seguirem o mesmo sentido de DH e DV, significa que o personagem está à direita ou acima do portal, então, utilizaremos sua normal invertida como multiplicador dos vetores diretores da câmera presente no outro lado.

6. Travessia

Para concluir, é necessário permitir que o personagem atravesse o portal de um lado para o outro. Isto é feito de forma simples: quando o jogador se aproxima de um portal, mudamos a sua posição para a frente do outro portal. A visão da câmera é modificada de modo a fazer com que o jogador esteja olhando na direção da normal do portal de saída. É importante notar que diversos objetos podem atravessar os portais, além do próprio jogador. Estes objetos podem ser utilizados para adicionar desafios em jogos que utilizam nossos portais.

Conclusão e trabalho futuro

Feitos todos estes passos, o produto que conseguimos é uma aplicação funcional que nos permite a transição entre portais renderizados e com perspectiva bastante verossímil.

Ainda há tarefas que podem ser implementadas para incrementar a ferramenta. Vamos expôr abaixo alguns pontos que são interessantes:

* Adição de física mais realista ao mundo
* Junto à adição de física, é importante permitir que o personagem seja capaz de sair de um portal no chão com mais suavidade
* Permitir que o personagem seja capaz de carregar objetos e carregá-los consigo através do portal
* Fazer com que a transição entre portais seja mais suave, sem que seja feito o teleporte; seria interessante que o personagem pudesse andar bem lentamente e até desistir no meio do caminho da transição, permitindo-lhe olhar com mais calma antes de realizar a travessia.