Relatório Detecção de Colisões

Victor Scherer Putrich

June 2020

1 Técnica de Subdivisão

Para este trabalho, optei por fazer a subdivisão de espaço para detectar colisão utilizando uma Quadtree que possibilita algumas modificações na subdivisão de espaço de forma dinâmica, permitindo que observasse algumas características da técnica que, na minha opnião, são mais interessantes que uma subdivisão uniforme.

1.1 Gerando a Quadtree

Na Quadtree é enviado por parâmetro o limite de linhas desejadas em um único subespaço, também a altura máxima que a quadtree deve possuir.

A Quadtree é inicializada apenas uma vez, criando um nodo raiz e chamando o método recursivo **Insert** que recebe um ponteiro pro vetor de linhas, geradas na inicialização da aplicação, e o número de linhas que se espera ter nesse ponteiro, a quadtree guarda um referencia para um nodo que vai armazenando o número total de linhas contidas naquele subespaço (o nodo raiz contém todas as linhas) e a medida que a subdivisão acontece, é recalculado o número de linhas que cada subespaço guarda, verificando se alguns dos pontos de extremidade das linhas está contida no subespaço ou se colide com alguma das bordas do mesmo, até que satisfaça o caso de parada por altura ou pelo número máximo de linhas, quando isso acontece, o nodo folha aloca espaço de memória suficiente para armazenar os índices das linhas que devem estar naquele subespaço, ou seja, cada folha da Quadtree tem o seu vetor de índices que referenciam as linhas do vetor de linhas.

1.2 Detecção de Colisão

Depois da construção da Quadtree, é necessário identificar a localização do veículo nos subespaços da quadtree, isso é feito através do método **Search** que faz um processo muito semelhante que a construção dos subespaços. O método recebe os pontos de extremidade do veículo e vai sendo identificado a que quadrante estes pontos pertencem ou se a linha formada a partir desses dois pontos passa por algum quadrante através da intersecção entre os limites do quadrante, o posicionamento do veículo para cada altura da árvore percorrida

vai ficando mais específico de forma que ao encontrar uma folha o nodo que a representa contém o vetor de índices das linhas que precisam ser testas com a colisão do veículo.

2 Observações na Construção da Solução

Quando o método Search é chamado, os procedimentos de colisão e coloração das linhas é feito durante a busca, inicialmente tinha optado por retornar o vetor de índices a ser testado, mas encontrei dificuldades em pensar em uma solução para agrupar os conjuntos de índices, afinal o veículo pode estar em mais de um subespaço.

Algumas construções das árvores demoram um pouco, em alguns casos não tinha sido possível construir a árvore, mas restringindo o número de subespaços através do limite de altura, esse problema é resolvido, porém, a construção ainda apresenta uma demora quando tem que construir árvores com muitos subespaços, principalmente com linhas grandes.

Fiquei satisfeito com o método de construção da árvore, me preocupei com a uso de memória em excesso, por isso, optei por alocar apenas o espaço necessário para armazenar os índices só nas folhas. Quando preciso guardar, por exemplo, o vetor de linhas na quadtree, estes são apenas ponteiros pro endereço do vetor original, uma possível otimização que farei é pensar numa forma de não repetir os índices de uma linha em mais de um subespaço, retornando um conjunto invés de um vetor.

3 Testes e Resultados

No código do trabalho, é possível fazer testes de colisão da maneira apresentada no código base ou ativar a Quadtree apertando 's', optei por modificar a forma que as linhas são instanciadas, alterando a geração de uma das cordenadas para ser igual a outra, garantindo que as linhas fiquem próximas a diagonal principal, este efeito só pôde ser verificado nas linhas curtas. O motivo é forçar a subdivisão de espaço de maneira mais irregular, para que a quadtree tenha au sua propriedade de subdivisão mais acentuada, caso contrário, seria uma subdivisão uniforme.

O ganho de desempenho é claro, até para quando o veículo se encontra dentro de vários subespaços, mas quando o veículos se encontra em um subespaço grande que contém poucas linhas, o ganho em comparação com o teste de colisão base é enorme. Notei que um número considerável de linhas pequenas (em torno de 10.000) ou um número um pouco menor de linhas grandes, subdividir espaços sem atingir um limite de altura, além de custar para construir a árvore, o ganho em throughput não existe em comparação com uma árvore de altura menor mas com mais linhas em cada subespaços, porque o veículo passa por vários quadrantes e os testes de colisão feitos para identificá-los acabam sendo o gargalo.

No teste de colisão com 1000 linhas já é claro que o desenho das linhas é o mais custoso para a detecção na Quadtree, a partir de 10000, com a remoção do desenho das linhas, existe um ganho de aproximadamento 100% no throughput.

O tamanho da linhas é um fator que contribui para o tempo de busca dos quadrantes que inicialmente, mesmo sem desenhar as linhas, tem um grande impacto na medição de desempenho, mas que com o aumento do número de linhas, a distancia do throughput de teste de colisão em Quadtree de linhas curtas e grandes, vai diminuindo.

Para a navegação do veículo nos quadrantes, existe uma diferença que para uma análise de desempenho é importante, mas que quando comparado ao teste de colisão global é pequena. Dependendo de onde o veículo está, o throughput oscila para mais ou menos (dentro dos testes que realizei).

Como mantive a subdivisão de espaço em 100 ou 500 linhas, mesmo que limitado a altura, um quadrante não passará tanto em seu número de linhas.

Foram feitos testes explorando as possibilidades que a Quadtree provê, fazendo uma análise na diferença entre árvores com altura limitada pelo número de linhas ou pela altura enviada por parâmetro, assim como a comporação entre a detecção de linhas curtas ou grandes.

Alguns testes não serão apresentados em tabela, mas que foram feitos para forçar os limites da Quadtree, junto com o hardware do meu computador. Gerei uma Quadtree com 100.000 linhas uma altura máxima de 8 níveis (4^8 subespaços) ou com subespaços de 100 linhas. Para os teste de colisão global, a taxa de frames por segundo foi, aproximadamente, de 0.035 e removendo os desenhos foi de 0.045 e para a Quadtree gerada foi de 0.2332 e removendo os desenhos foi de 0.93895. Outros testes, com um menor número de linhas, mas que não foram tabelados por ainda não ter adotado um critério de avaliação, apresentaram até 1000% de aumento throughput.

A figura 1 mostra alguns dos resultados obtidos:

Tipo Linha	Max Linhas	Tipo Intersec	Max Linhas Subespaço	Altura Max Quadtree	Linha Ligada	Throughput
curta	100	GLOBL	0	0	1	33,33333
curta	100	GLOBL	0	0	0	38,4615
curta	100	QUAD	10	10	1	166.667
curta	100	QUAD	10	10	0	250
grande	100	GLOBL	0	0	1	31,25
grande	100	GLOBL	0	0	0	35,7143
grande	100	QUAD	10	10	1	166.667
grande	100	QUAD	10	10	0	500
curta	1000	GLOBL	0	0	1	3,3557
curta	1000	GLOBL	0	0	0	4,42478
curta	1000	QUAD	50	10	1	23,8095
curta	1000	QUAD	50	10	0	71,4286
grande	1000	GLOBL	0	0	1	3,06748
grande	1000	GLOBL	0	0	0	3,78788
grande	1000	QUAD	50	10	1	18,5185
grande	1000	QUAD	50	10	0	55,5556
curta	10000	GLOBL	0	0	1	0,344353
curta	10000	GLOBL	0	0	0	0,455373
curta	10000	QUAD	100	4	1	3,08642
curta	10000	QUAD	100	4	0	9,25926
curta	10000	QUAD	500	10	1	2,90698
curta	10000	QUAD	500	10	0	7,69231
grande	10000	GLOBL	0	0	1	0,338735
grande	10000	GLOBL	0	0	0	0,425532
grande	10000	QUAD	100	10	1	1,16009
grande	10000	QUAD	100	10	0	7,93651
grande	10000	QUAD	500	4	0	7,46269

Figure 1: Tabela de Resultados