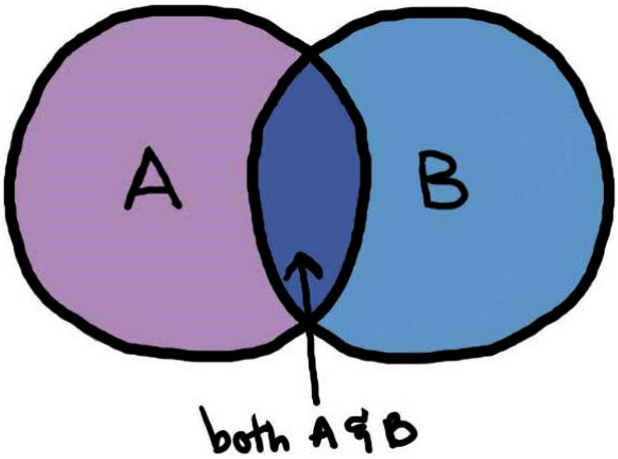
Tutorial de Física 5: Variedades de Colisão



Resumo

Neste tutorial, expandiremos nosso procedimento de detecção de colisão para gerar com precisão uma variedade de colisão. Estaremos cobrindo o que implica uma variedade de colisão, juntamente com uma discussão sobre o método de recorte que se tornará o principal método de calcular a variedade de colisão nesta série de tutoriais.

Novos Conceitos

Pontos de contato, variedade de colisão, método de recorte, recorte Sutherland-Hodgman

Introdução

Neste ponto, identificamos quando dois objetos colidiram e recuperamos o normal de colisão/contato$N$bem como a distância de penetraçãop. No entanto, mais uma informação é necessária antes que nosso mecanismo de física possa passar para o último estágio de seu ciclo de atualização e realmente resolver nossas colisões.

Especificamente, precisamos identificar os pontos de contato. Anteriormente, considerávamos o contato*apontar* como se aplica na detecção de colisões (e na resposta a colisões, discutida em um tutorial futuro), mas uma abordagem simples é inadequada para resolver colisões sofisticadas.

Neste tutorial apresentaremos o coletor de colisão. Definiremos sua finalidade e explicaremos como ele pode ser calculado usando o Método Clipping. Ao final deste tutorial, teremos todas as informações necessárias para realizar atualizações de resposta a colisões, que serão o assunto dos dois tutoriais finais da parte de física deste módulo.

O que é o Ponto de Contato?

No momento temos a direção da colisão (normal) e a distância de penetração. No entanto, se isso fosse tudo o que fosse usado para resolver uma colisão entre dois objetos, então nenhuma rotação ocorreria em nosso mecanismo de física. Lembramos que existe outro dado de colisão: o ponto de contato.

Um ponto de contato descreve um ponto no qual dois objetos se tocam. Isso pode ser usado para resolver colisões na forma de uma restrição de distância, para impedir que dois objetos se sobreponham no intervalo de tempo seguinte.

Deveria ser óbvio, entretanto, que mesmo um ponto de contato não transmitirá necessariamente todas as informações necessárias para gerar rotações significativas em resposta a uma colisão. Considere a diferença entre uma moeda rolando em uma superfície e um pneu. Se quisermos tentar responder com precisão às colisões detectadas, precisaremos coletar mais dados sobre como nossos objetos estão interagindo - é aqui que a variedade de colisões é útil.

O que é um coletor de colisão?

Uma variedade de colisão é uma coleção de pontos de contato que formam todas as restrições necessárias que permitem ao objeto resolver adequadamente todas as penetrações. Pode ser visto como a soma da área de superfície entre dois objetos em colisão. Conforme mostrado na Figura 1, isso pode formar um único ponto, uma linha ou um polígono 2D.

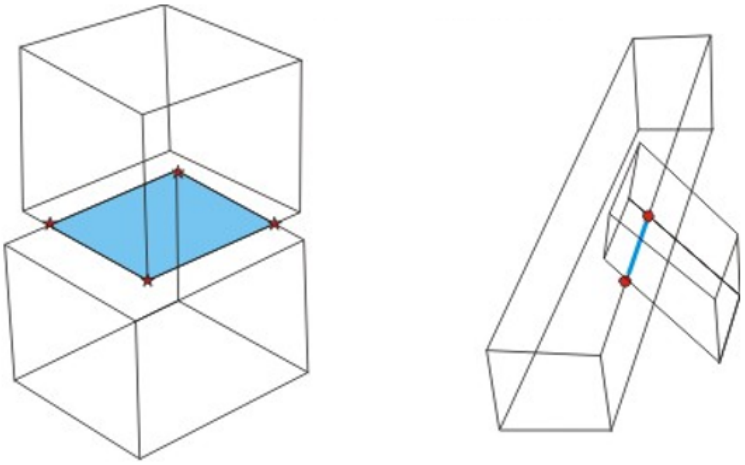


Figura 1: O coletor de contato

Em um sistema físico discreto, isso representa um problema, pois as colisões só são detectadas depois que os dois objetos já estão sobrepostos. Isto resulta não numa área de superfície 2D onde os dois objetos se tocam, mas sim num volume 3D pelo qual eles já se interpenetram.

Para superar isso, inferimos a variedade de contato como se os dois objetos estivessem apenas se tocando. Isso nos permite lidar com a resolução da colisão como se fosse um evento real (já que a interpenetração não ocorre nas ocorrências do mundo real das colisões que estamos modelando).

O método de recorte

Para calcular a variedade usaremos o método de recorte, no qual recortaremos progressivamente uma face de um objeto com o perímetro de um segundo objeto. Isso resulta em uma variedade de colisão 2D que pode então ser usada em nossos cálculos de resolução.

A melhor forma de mostrar como esse algoritmo funciona é através de um exemplo. Considere o cenário mostrado na Figura 2.

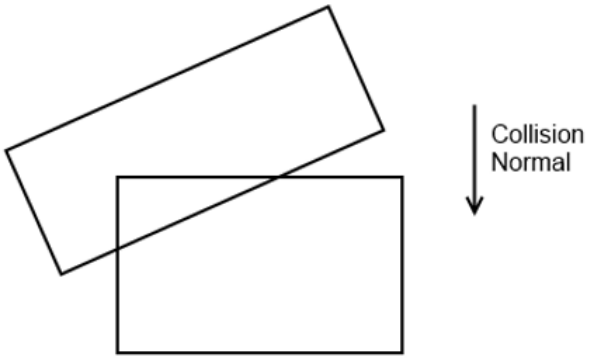


Figura 2: Cenário de Colisão

Neste caso hipotético, duas caixas colidiram. Neste ponto de nossa execução, acabamos de executar nossas rotinas SAT e conhecemos tanto a colisão normal N e profundidade de penetração p.

Existem várias etapas para determinar a variedade através do método de recorte, e abordaremos cada uma delas, começando com o processo pelo qual identificamos faces significativas (aquelas envolvidas na colisão).

Identificando as Faces Significativas

O primeiro passo é identificar as faces significativas que estão se cruzando. Isto é conseguido selecionando o vértice mais distante ao longo da normal de colisão. Na Figura 3, esses vértices estão destacados com círculos vermelhos.

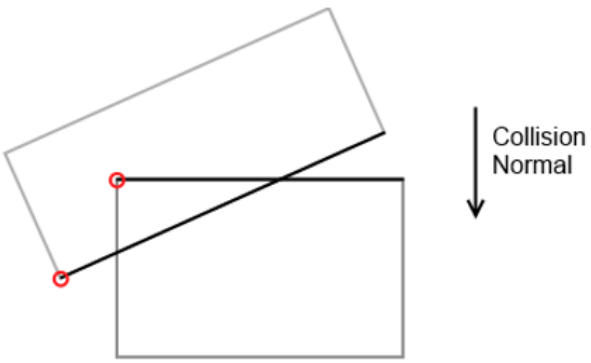


Figura 3: Vértices mais distantes ao longo da normal de colisão

A seguir selecionamos uma face em cada objeto que satisfaça os seguintes critérios:

- A face inclui o vértice selecionado
- A normal da face é a mais próxima do paralelo com a normal de colisão de todas as faces que contêm o vértice selecionado

Fazer isso para ambos os objetos nos dá as duas faces mais significativas para geração de contato.

Observação: A normal é invertida ao selecionar o vértice do segundo objeto.

Calculando o Incidente e as Faces de Referência

A face de referência se tornará o ponto de referência quando o recorte ocorrer nas etapas subsequentes da verificação. A face incidente, por sua vez, se tornará um conjunto de vértices que serão recortados.

Para fazer isso, calculamos qual das duas faces significativas tem uma normal mais próxima do paralelo com a normal de colisão. Considere a Figura 4. Neste caso a normal da face indicada por uma linha azul está mais próxima do paralelo e, como tal, essa face torna-se a face de referência.

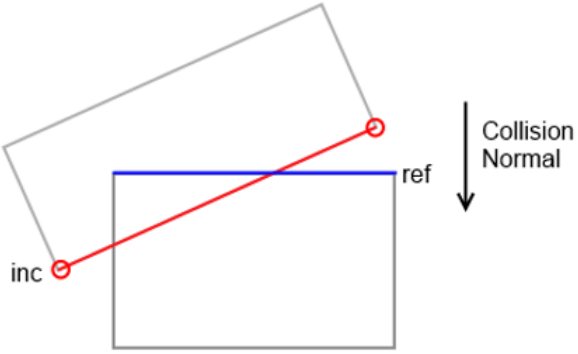


Figura 4: O coletor de contato

A outra face torna-se então a face incidente que iremos recortar para gerar os pontos de contato. No nosso exemplo, é composto por dois vértices.

Recorte de rosto adjacente

Agora recortamos o incidente com todas as faces adjacentes da referência. Isso é feito tomando as faces adjacentes normais e qualquer vértice que ela contenha para produzir uma equação plana. O algoritmo que usamos para calcular o recorte é conhecido como Sutherland-Hodgman Clipping. Isto pode ser facilmente adaptado para se adequar a um cenário 3D, tornando-o apropriado para uso em nosso mecanismo de física.

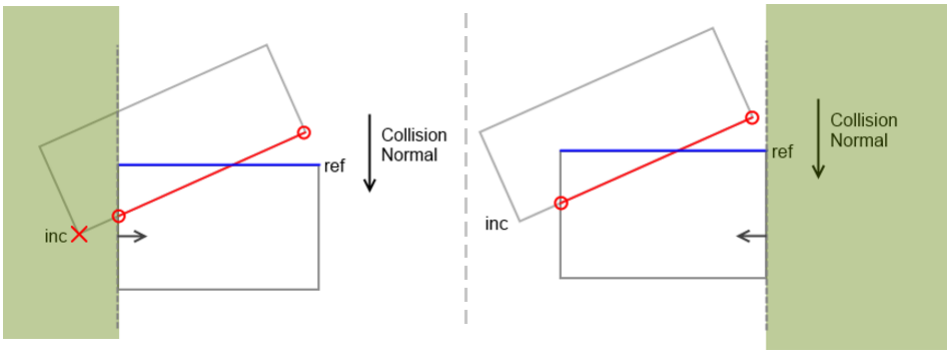


Figura 5: Recorte de Face Adjacente

O primeiro plano de recorte que ilustraremos neste exemplo é a face esquerda, mostrada no lado esquerdo da Figura 5. Como um dos vértices da face incidente está dentro da região de recorte, ele será substituído por um vértice que fica na borda do plano de recorte. O segundo plano de recorte é o da face direita, mostrado no lado direito da Figura 5. Neste caso, deve-se observar que nenhum dos pontos da face incidente está na região de recorte, portanto nenhuma alteração será feito.

Recorte Final

O plano de corte final é o da própria face de referência. Entretanto, em vez de recortar a face do incidente como no estágio anterior, agora apenas removemos todos os pontos que estão dentro da região de recorte. Como mostrado na Figura 6, isso nos deixa com apenas um único ponto de contato e não uma linha ou polígono.

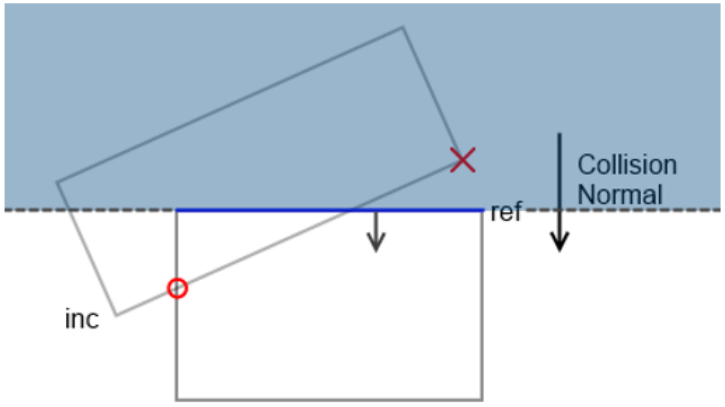


Figura 6: Recorte da Face Adjacente - Face Direita

Embora à primeira vista pareça que estamos a ignorar pontos de contacto críticos, isto é de facto correcto. O que estamos tentando inferir são os pontos de contato quando os dois objetos se tocaram pela primeira vez, e nem todos ocorreram desde que se sobrepuseram. Neste exemplo, apenas o canto da face de referência estaria em contato com o outro objeto; isso é óbvio quando comparamos as formas com a direção de deslocamento ao longo da normal de colisão.

Isto pode parecer um desperdício de verificação, dado que o coletor produzido é apenas um único ponto de contato. É importante lembrar, porém, que a variedade de colisão pode ser um ponto, dois pontos ou muitos; a razão pela qual realizamos o processo de recorte é para obter a ideia mais precisa possível da variedade de colisão, independentemente do número de pontos gerados. Se considerarmos a natureza das colisões entre objetos convexos em geral, deveria ser óbvio que a maioria das variedades de colisão serão apenas um único ponto - pois representam a maneira pela qual os objetos começaram a interagir; o método de recorte nos permite resolver os cenários mais complexos onde este não é o caso.

Implementação

Revise o dia 4 da apostila de Tarefas Práticas. Tente aproveitar o tempo adicional esta tarde para estender sua abordagem de detecção de colisão para se adequar a objetos ainda mais complexos. Se você tiver a oportunidade, considere estender sua abordagem para seleção de fases amplas.

Resumo do tutorial

Neste tutorial, introduzimos o conceito de variedade de colisão e explicamos sua importância na obtenção de respostas de colisão confiáveis, particularmente no contexto do movimento angular. Determinamos passo a passo como extrair a variedade de colisão de uma maneira eficiente e facilmente adaptável para se adequar ao nosso mecanismo físico. Agora temos todos os dados de colisão necessários para implementar o estágio final do nosso ciclo de atualização física: resposta à colisão.

```

//MotorFísico ::NarrowPhaseCollisions()

//    Depois:
      boolokA =
            cp.pObjectA ->FireOnCollisionEvent(cp.pObjectA , boolok B =         cp.pObjectB );

            cp.pObjectB ->FireOnCollisionEvent(cp.pObjectB ,                    cp.pObjectA );

//Inserir:

se(okA && okB)
{
      // Construa um coletor de colisão completo que também tratará a // resposta de
      colisão entre os dois objetos no // estágio do solucionador


      Coletor* coletor =novoMúltiplo ();

      coletor ->Iniciar(cp.pObjectA , cp.pObjectB );

      // Construa pontos de contato que formam o perímetro da // variedade de colisão


      colDetect.GenContactPoints(manifold);

      se(coletor ->contactPoints.size() > 0) {

            // Adiciona à lista de variedades que precisam ser resolvidas
            coletores.push_back(coletor);
      }
      outro
            excluirmúltiplo;
}
```

PhysicsEngine.cpp

```

//CollisionDetectionSAT::GenContactPoints()

se   (!out_manifold || !areColliding) retornar;


se(bestColData._penetração >= 0,0f)
      retornar;

// Obtém as informações de face necessárias para as duas formas ao redor da // normal de
colisão

std::list <Vetor3 > polígono1 , polígono2; Vetor3
normal1, normal2;
std::vector <Plano > adjPlanes1 , adjPlanes2;

cshapeA ->GetIncidentReferencePolygon(
      bestColData._normal , polygon1 , normal1 , adjPlanes1 );

cshapeB ->GetIncidentReferencePolygon(
      - bestColData._normal , polygon2 , normal2 , adjPlanes2 );
```

```
22
23 // Se shape1 ou shape2 retornaram um único ponto, então ele deve //   estar em uma curva e,
24 portanto, o único ponto de contato a ser gerado // já está disponível
25
26
27 se(polygon1.size() == 0 || polygon2.size() == 0) {
28
29       retornar;// Nenhum ponto retornado, resultando em nenhum contato possível
30                 // pontos
31 }
32 senão se(polígono1.tamanho() == 1) {
33
34       out_manifold ->AdicionarContato(
35             polígono1.front(),// Polígono1 -> Polígono 2
36             polygon1.front() + bestColData._normal
37             * bestColData._penetration , bestColData._normal ,
38             bestColData._penetration );
39 }
40 senão se(polígono2.tamanho() == 1) {
41
42       out_manifold ->AdicionarContato(
43             polygon2.front() - bestColData._normal
44             * bestColData._penetration , polygon2.front(),// Polígono2
45             <- Polígono 1 bestColData._normal ,
46
47             bestColData._penetração);
48 }
49 outro
50 {
51       // Caso contrário, use o recorte para cortar a face incidente para ajustá-la // dentro dos
52       planos de referência usando os planos da face circundante
53
54       // Primeiro precisamos saber se precisamos inverter o incidente e fazer referência // aos
55       rostos para recorte
56
57       boolinvertido = fabs(Vector3 ::Dot(bestColData._normal , normal1 ))
58             < fabs(Vector3 ::Dot(bestColData._normal , normal2 ));
59
60       se(virado)
61       {
62             std::swap(polygon1 ,           polígono2);
63             std::swap(normal1 ,          normal2);
64             std::swap(adjPlanes1 ,              adjPlanes2 );
65       }
66
67       // Recorta a face incidente nas arestas adjacentes da referência // face
68
69
70       se(adjPlanes1.size() > 0)
71             SutherlandHodgmanClipping(polygon2 , adjPlanes1.size(),
72                   &adjPlanes1 [0], &polígono2 ,falso);
73
74       // Finalmente recorte (e remova) quaisquer pontos de contato que estejam acima // da
75       face de referência
76
77       Plano refPlane =
78             Plano(-normal1 , -Vector3 ::Dot(-normal1 , polygon1.front ()));
79       SutherlandHodgmanClipping(polígono2 , 1, &refPlane , &polígono2 ,verdadeiro);
```

```
80
81      // Agora ficamos com uma seleção de pontos de contato válidos a serem // usados   para o
82      coletor
83
84      para(constVetor3 e ponto: polígono2) {
85
86          //Calcula a distância até o plano de referência
87
88          Vetor3       pontoDiff =
89              apontar   - GetClosestPointPolygon(ponto,                  polígono1);
90          flutuadorpenetração_de_contato      =
91              Vetor3 ::Ponto(pontoDiff ,            bestColData._normal );
92
93          //Definir dados de contato
94
95          Vetor3 globalOnA = ponto; Vetor3
96          globalOnB =
97              ponto - bestColData._normal * contact_penetração;
98
99          // Se invertermos os planos de incidente e de referência, // precisaremos invertê-
100         lo antes de enviá-lo ao coletor. // por exemplo, deixe de falar sobre object2
101         ->object1 em // object1 ->object2
102
103
104         se(virado)
105         {
106             penetração_de_contato       =   - penetração_de_contato;
107             globalOnA =
108                 ponto + bestColData._normal * contact_penetração;
109
110             globalOnB = ponto;
111         }
112
113         // Apenas faça uma verificação final de sanidade se o ponto de contato // é
114         realmente um ponto de contato e não apenas um bug de recorte
115
116         se(penetração_de contato <0,0f) {
117
118             out_manifold ->AdicionarContato(
119                 globalOnA ,
120                 globalOnB,
121                 bestColData._normal ,
122                 penetração_contato);
123         }
124     }
125 }
```

CollisionDetectionSAT.cpp