PEF – 5743 – Computação Gráfica Aplicada à Engenharia de Estruturas

Prof. Dr. Rodrigo Provasi

e-mail: provasi@usp.br

Sala 09 – LEM – Prédio de Engenharia Civil

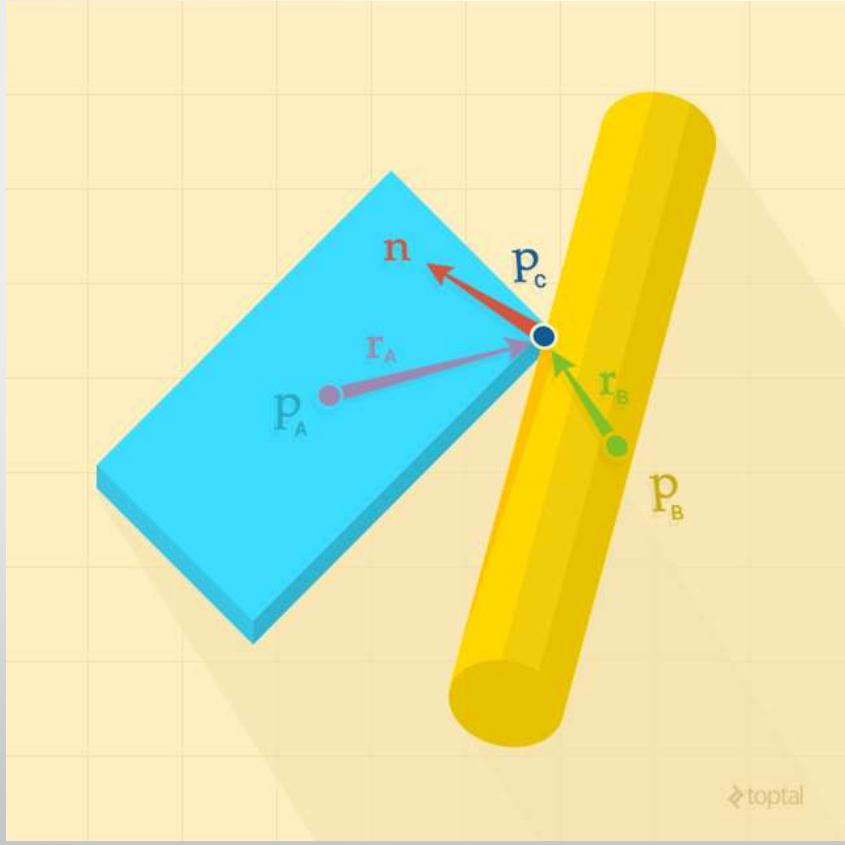
Colisão

- Durante a execução de um programa de computação gráfica, os elementos que compõem a cena não estão estáticos
- Essa movimentação não é previamente determinada, já que pode depender da entrada do usuário ou de algum cálculo executado internamente.
- Pode haver colisão!

Colisão

- O tratamento da colisão muda o comportamento dos sólidos de tal forma que fique mais realista, já que no tratamento do contato pode-se aplicar forças, mudar a direção do movimento e das forças.
- Colisão de objetos rígidos.
  - Eles não se deformam.
  - Não há a necessidade de se recalcular os pontos da geometria.

Colisão



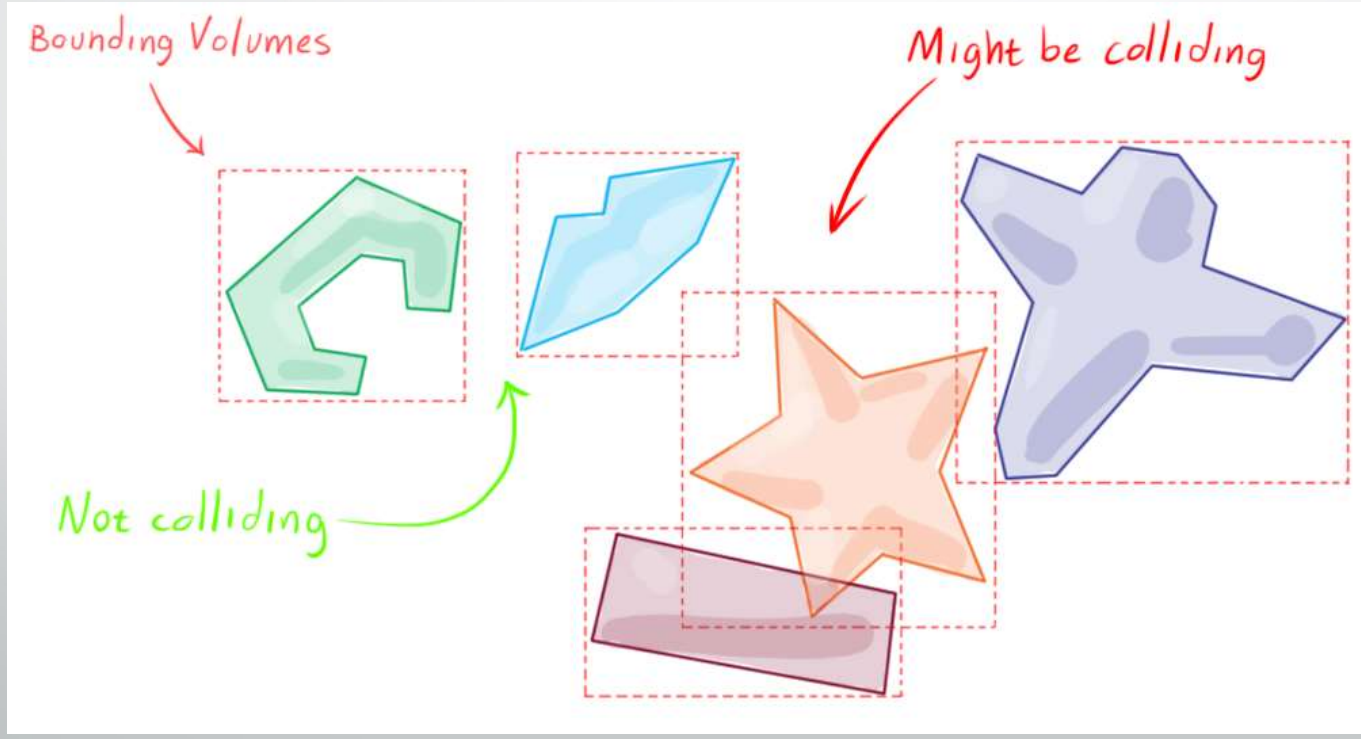
Colisão

- A colisão entre dois corpos acontece quando os eles se interseccionam ou quando a distância entre eles é menor que uma tolerância.
- Se existir n corpos na cena, a complexidade em detectar colisões é um teste par a par cuja complexidade é da ordem de n^2.
- Essa verificação é custosa computacionalmente, já que os corpos podem estar em posições e orientações arbitrárias.
- Para que o processo seja otimizado, normalmente divide-se o processo de detecção em duas fases: uma mais ampla e outra mais específica.

Colisão

- A primeira fase faz uma busca ampla buscando pares que potencialmente estão colidindo e excluindo os pares que certamente não entrarão em contato.
- Essa fase tem que ser eficiente a tal ponto para não incorrer no mesmo custo computacional quadrático anteriormente mencionado.
- Para que isso seja alcançado utiliza-se abordagens que particionam o espaço com volumes limitantes (*bounding volumes*), que ajudam estabelecer limites superiores para a colisão.

Colisão



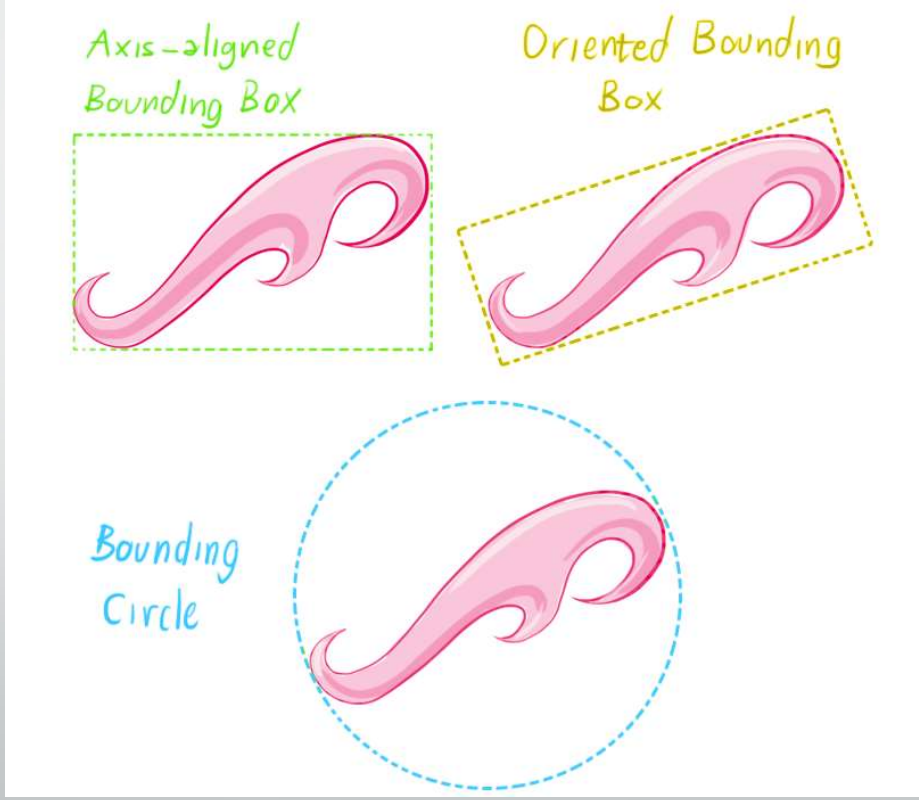
Fase Ampla

- Na fase ampla da colisão é preciso identificar os pares de corpos rígidos que podem estar em contato.
- Esses corpos podem ter formas complexas como polígonos e poliedros.
- Para acelerar as buscas, podemos utilizar uma forma mais simples que envolva o corpo em questão. Essa envoltória é conhecida como *bounding boxes* ou *bounding volumes*.

Fase Ampla

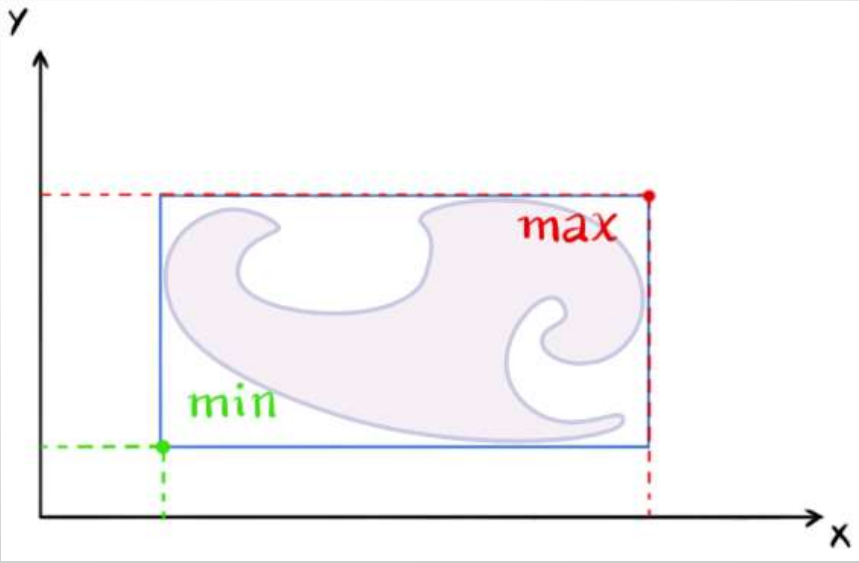
- Se essas envoltórias não se intersectarem, os corpos definitivamente não estão em contato; já se por acaso houver intersecção entre as envoltórias, pode haver contato e o conjunto é passado para a fase mais específica.
- Alguns dos possíveis *bounding volumes* que são mais comumente utilizados são os *Oriented Bounding Boxes (OBB)*, círculos no 2D e esferas no 3D. Uma outra possibilidade é a *Axis-Aligned Bounding Boxes (AABB)*. Como o nome diz, a caixas envoltórias está alinhada com os eixos.

Fase Ampla


Axis-aligned
Bounding Box
Oriented Bounding
Box
Bounding
Circle

Fase Ampla

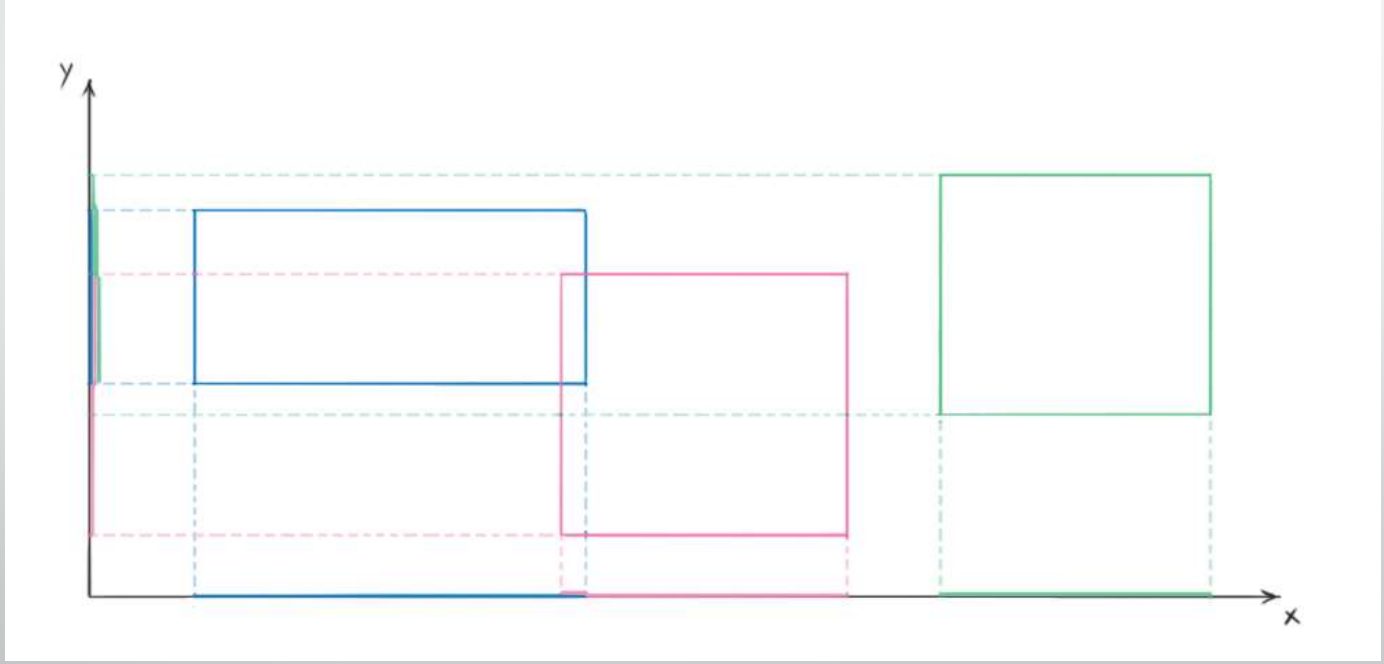
- *Bounding box* do tipo AABB é muito utilizada por ser de fácil definição. Uma vez estabelecido os pontos mínimo e máximo, a região está completamente delimitada.



Fase Ampla

- Para verificar se duas envoltórias descritas com esse método se intersectam, verifica-se se a projeção das projeções das envoltórias nos eixos se intersectam.
- Para haver colisão, todas as projeções devem se intersectar (ou seja, as interseções devem ocorrer nos dois eixos no espaço 2D e nos três eixos no espaço 3D).

Fase Ampla



Fase Ampla

- Embora esse teste seja pouco custoso computacionalmente, não há ganhos se todos os possíveis pares terão que ser testados, já que a complexidade continua da ordem de n^2.
- Para minimizar o número de testes necessários, algum tipo de partição espacial deve ser utilizado.
- Essas partições seguem os mesmos princípios de banco de dados indexados que acelera as buscas.

Fase Ampla

- Como as envoltórias se movem no decorrer a execução do programa, os índices devem ser recalculados em cada passo da simulação / animação.
- Existem diversos tipos de algoritmos de partição e estrutura de dados que podem ser utilizados como por exemplo:
  - grades uniformes (*uniform grids*)
  - *quadtrees* (2D)
  - *octress* (3D)
  - *hashing* espacial

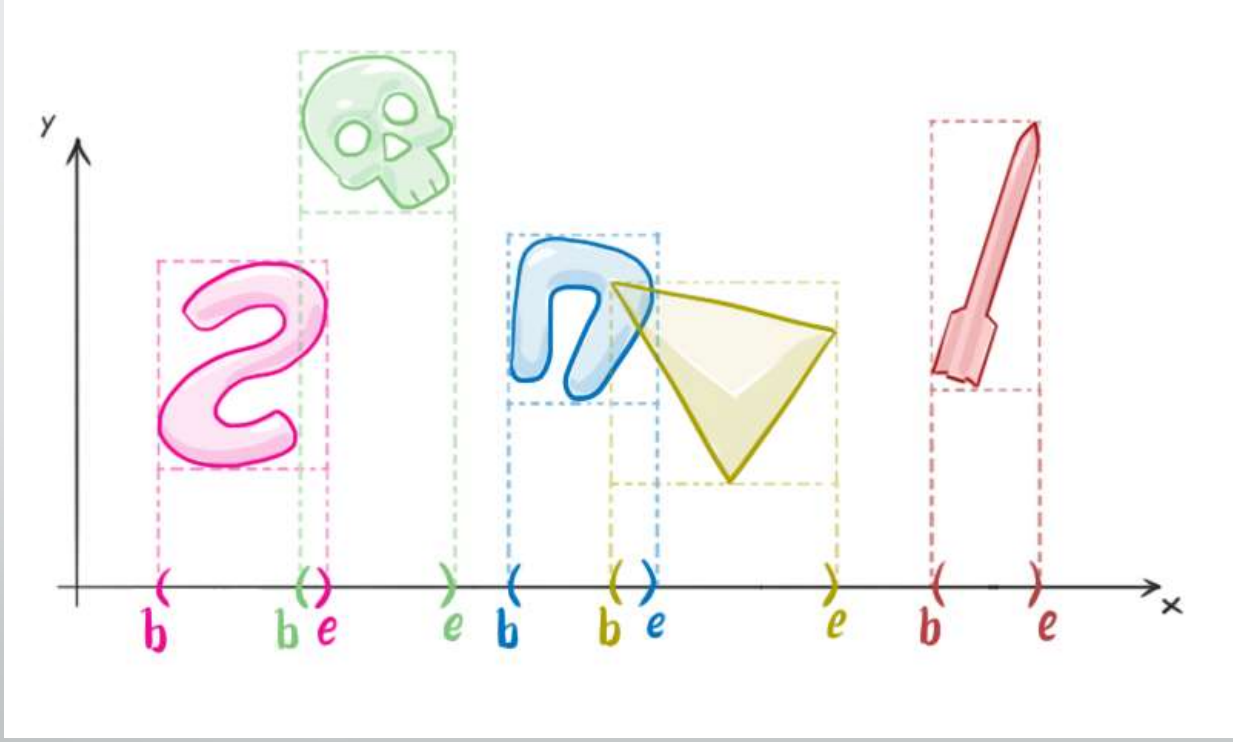
Fase Ampla

- A seguir alguns dois algoritmos espaciais serão exibidos: *sort and sweep* (ordenar e varrer) e *Bounding Volume Hierarchies* (BHV – Envoltória de volumes hierárquicas).

Sort and Sweep

- O método de ordenar e varrer é em geral o escolhido para ser programado quando já contato entre corpos rígidos. Considere a projeção de uma envoltória do tipo AABB em um único eixo de coordenada.
- Isso é essencialmente um intervalo [b,e] com b sendo o início e e o fim.
- O que se deseja saber é quais intervalos se interseccionam.

Sort and Sweep



Sort and Sweep

- Nesse algoritmo insere-se todos os valore de b e e numa única lista e se ordena de forma ascendente pelos valores escalares.
- Em seguida, vare-se a lista. Quando um valor b é encontrado, o intervalo correspondente é separado em outra lista de intervalos ativos e, quando um valor e é encontrado, o intervalo é removido da lista de intervalos ativos.
- Em um determinado momento, todos os intervalos ativos estão interseccionando.
- Essa lista de intervalos pode ser reutilizada para cada passo da simulação, de tal forma a tornar a reordenação da lista de maneira mais eficiente.

Sort and Sweep

- Rodando esse algoritmo, não importando se o problema é em duas ou três dimensões, o número de testes de interseção que precisam ser executados é muito menor, já que a verificação acontece em apenas um dos eixos.
- Existem versões mais sofisticadas desse algoritmo que tornam ainda mais rápidas a análise e, consequentemente, diminuem o custo da busca de possíveis elementos que colidam.

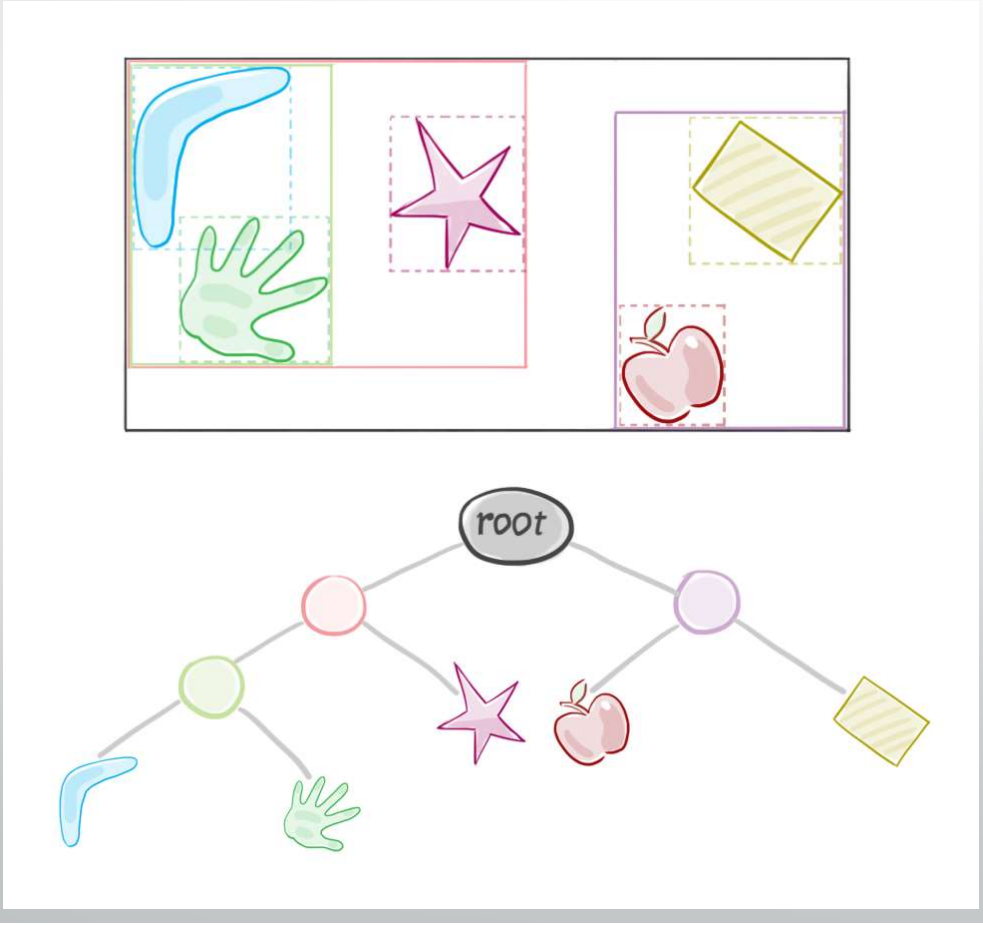
Árvores dinâmicas de envoltórias volumétricas

- Um método de partição especial bem utilizado é o de árvores dinâmicas de envoltórias volumétricas (ou *Dynamic Bounding Volume Tree* ou DBVT). Essa abordagem cria uma hierarquia de volumes.
- O DBVT é uma árvore binária na qual cada nó é uma AABB que circunda todas as AABBs dos filhos.

Árvores dinâmicas de envoltórias volumétricas

- Essas envoltórias estão localizadas nas folhas da árvore.
- Tipicamente, essa estrutura pode ser consultada e permite detectar as colisões.
- A operação é eficiente porque não há necessidade de verificar os nós filhos das envoltórias que estão em outro nó da árvore.
- A colisão inicia-se do nó raiz e continua recursivamente até encontrar as envoltórias que se interseccionam.

Árvores dinâmicas de envoltórias volumétricas



Fase específica

- Depois que a fase ampla detecta os pares que podem potencialmente entrar em contato passa-se para a fase específica.
- Nela, para cada par, verifica-se se os corpos estão de fato colidindo, intersectando ou a distância entre eles é menor que uma tolerância, considerando forma posição e orientação do corpo.
- Precisa-se também verificar qual o ponto (ou pontos) de contato .

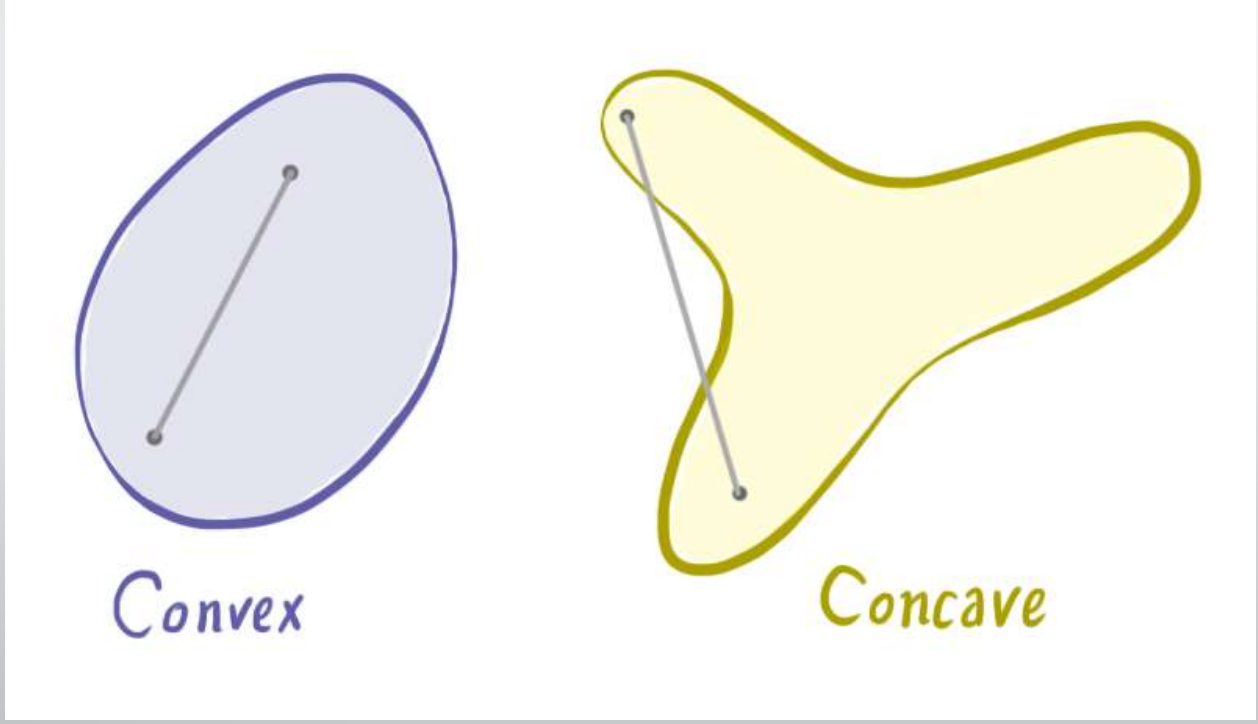
Figuras côncavas e convexas

- Determinar se duas figuras de formas arbitrárias estão próximas, em contato ou interseccionando não é uma tarefa trivial.
- Entretanto, uma propriedade que é extremamente importante é a concavidade da forma.
- Essas formas podem ser côncavas e convexas.

Figuras côncavas e convexas

- As formas côncavas são mais difíceis de serem trabalhadas e algumas estratégias foram desenvolvidas para lidar especificamente com esse problema.
- Numa figura convexa, uma linha que ligue quaisquer dois pontos da figura fica completamente dentro da figura.
- Entretanto, para uma figura côncava (ou não convexa), a mesma regra não é verdade, ou seja, pelo menos um segmento ligando dois pontos do corpo passa por fora da figura.

Figuras côncavas e convexas



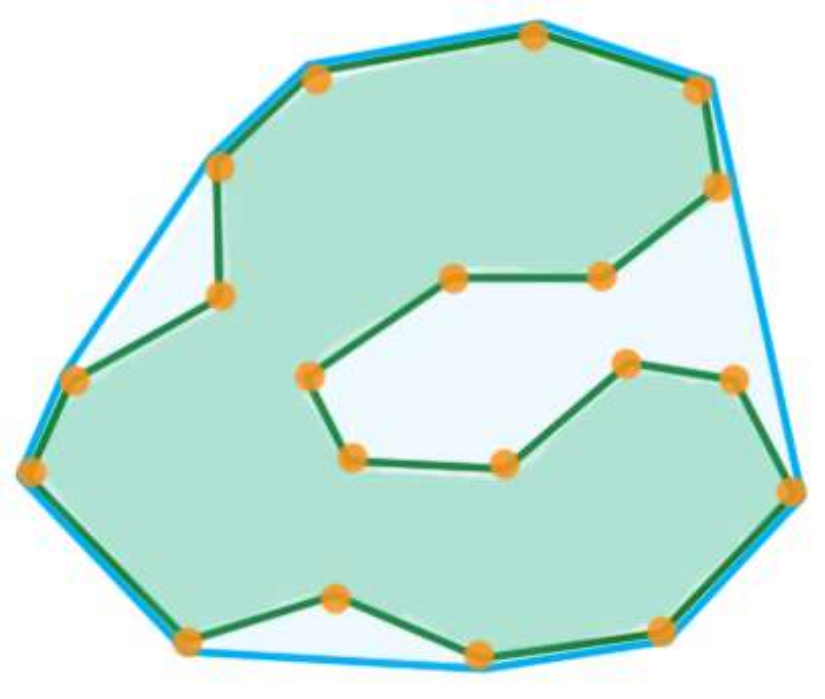
Figuras côncavas e convexas

- Computacionalmente, deseja-se que todas as figuras sejam convexas, pois existem disponíveis uma grande gama de algoritmos que determinam distância e intersecção e que funcionam apenas com figuras convexas.
- Como nem todos os objetos são convexos, procura-se contornar esse problema através de duas técnicas: *convex hull* (ou casco convexo) e decomposição convexa.

Convex Hull

- O *convex hull* de uma figura é a menor figura convexa que contém a figura por inteiro.
- Pode-se imaginar que é um polígono formado por um elástico preso ao redor da figura com os pontos do polígono sendo pregos que mantém esse elástico no lugar.
- Existem algoritmos que permitem calcular essa envoltória e que apresentam um custo computacional da ordem $n \log n$.

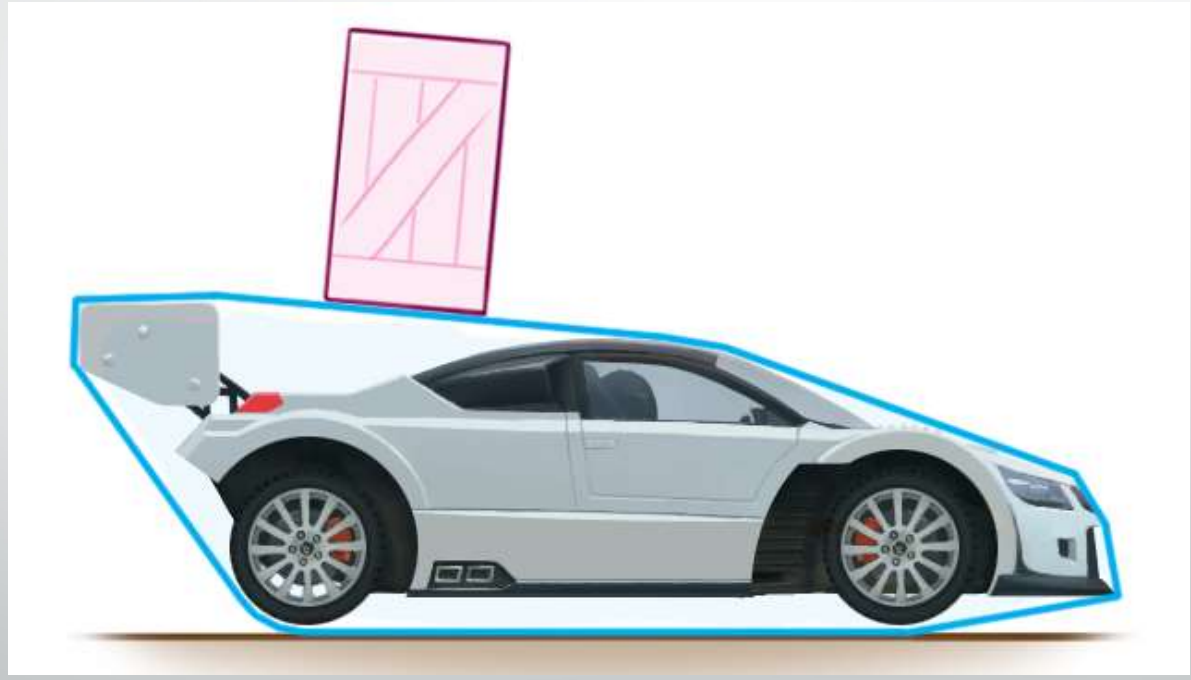
Convex Hull



Convex Hull

- Evidentemente, quando usa-se essa abordagem, perde-se as características côncavas do objeto.
- Isso pode levar ao que se denomina colisões “fantasma”, já que o objeto ainda será representado pela figura côncava.

Convex Hull



Convex Hull

- Em geral, a abordagem do *convex hull* é suficiente para representar objetos côncavos, uma vez que as colisões irreais são pouco visíveis ou as propriedades côncavas não são importantes para a eventual simulação.
- Porém, em alguns casos, necessita-se que o objeto seja representado de maneira realista.
- Por exemplo, se utilizar-se a abordagem do *convex hull* para representar uma tigela, seria impossível colocar algo nela; tudo ficaria por cima da tigela.
- Nesses casos, a decomposição convexa se mostra mais interessante.

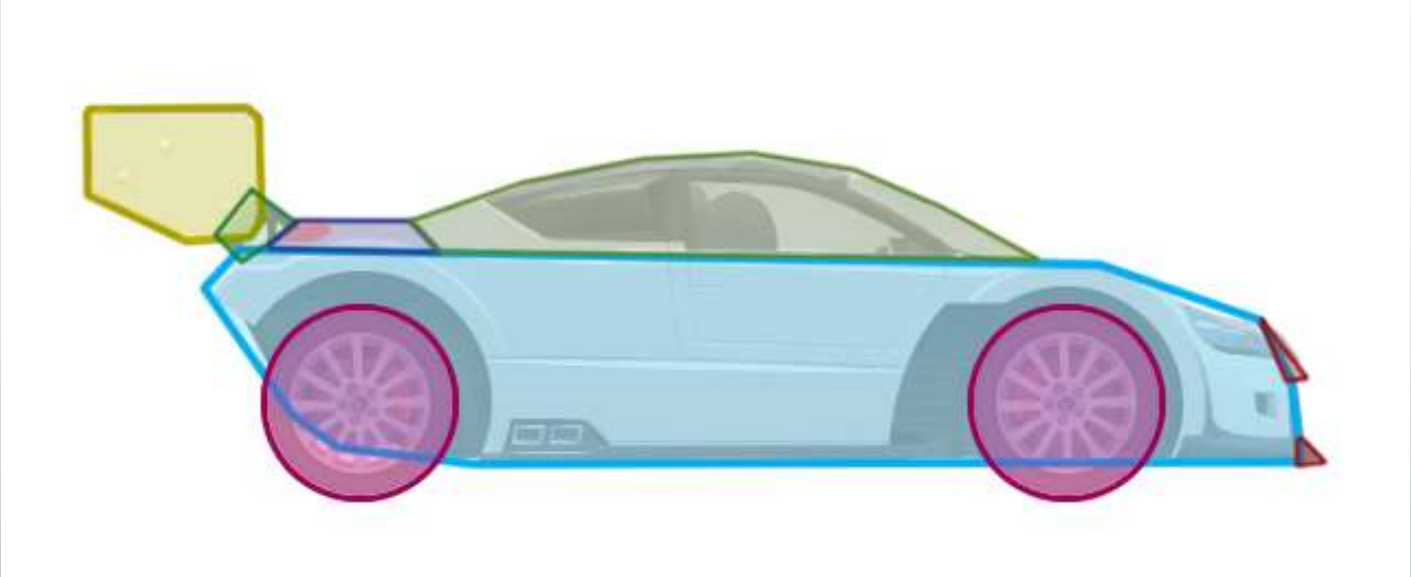
Decomposição Convexa

- O algoritmo de decomposição convexa recebe um objeto côncavo e retorna um conjunto de figuras convexas cuja união é a figura original.
- Algumas figuras só podem ser com um grande número de figuras convexas e isso pode levar a um grande custo computacional.

Decomposição Convexa

- Em vários problemas físicos que envolvam contato a decomposição convexa pode ser feita manualmente.
- Isso elimina a necessidade de utilizar algoritmos de decomposição.
- Como a performance é um dos aspectos mais importantes em simulações em tempo real, normalmente se utiliza uma geometria simplificada.

Decomposição Convexa



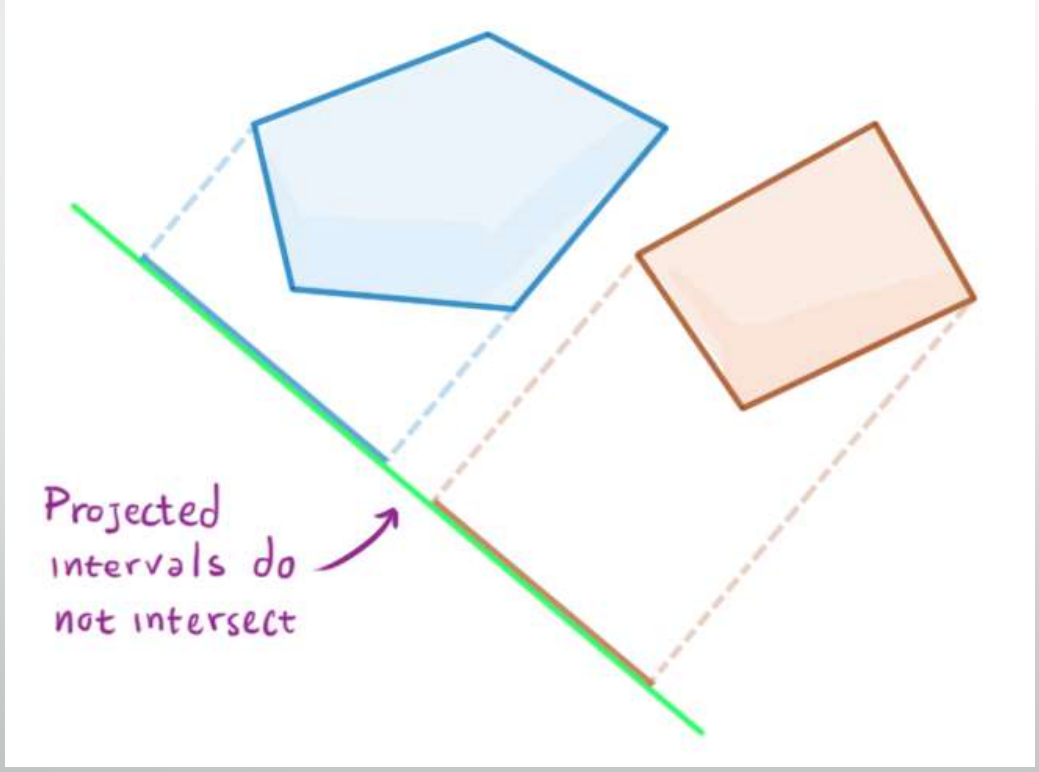
Testes para determinar intersecção

- Para determinar a interseção precisa-se compreender o teorema do eixo separador.
- Para determinar a distância entre os corpos, utiliza-se o algoritmo Gilbert-Johnson-Keerthi.
- Ambos serão vistos em detalhes a seguir.

Teorema do eixo separador

- O teorema do eixo separador (*SAT* do inglês *Separating Axis Theorem*) diz que duas figuras convexas não interseccional se, e somente se, existir pelo menos um eixo tal que as projeções ortogonais das figuras não se interseccionam.

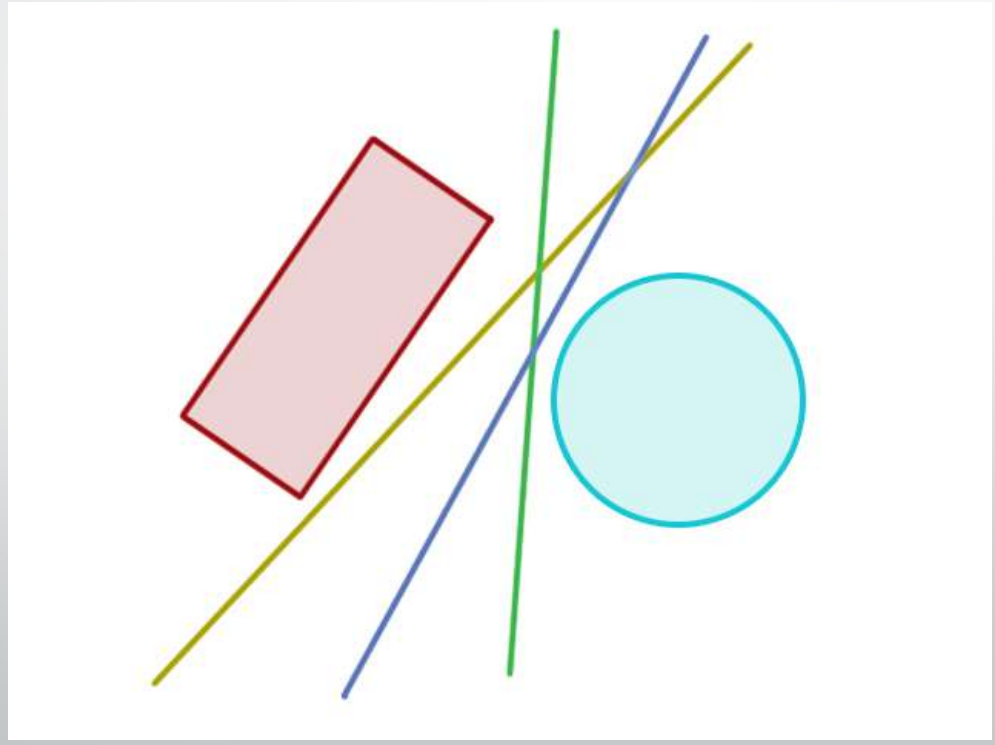
Teorema do eixo separador



Teorema do eixo separador

- É mais intuitivo achar uma linha no 2D ou um plano no 3D que separe as figuras, o que é efetivamente o mesmo princípio. O vetor ortogonal a essa linha 2D ou normal ao plano 3D pode ser utilizada como eixo separador.

Teorema do eixo separador



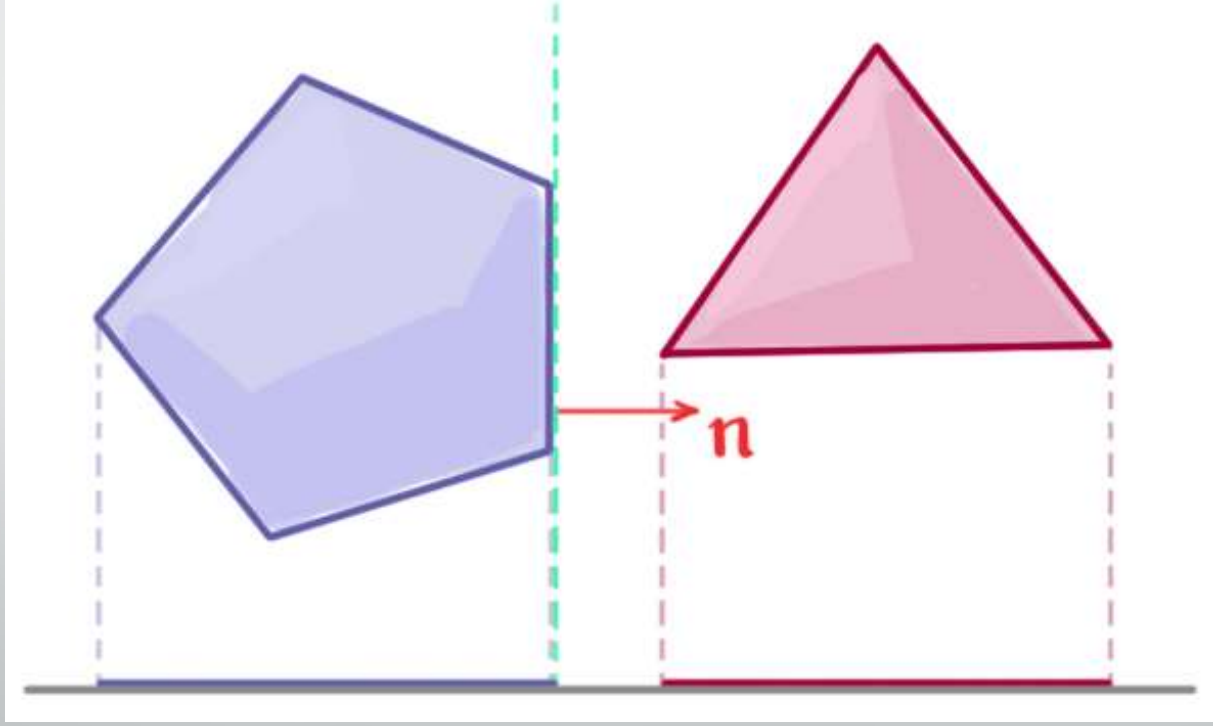
Teorema do eixo separador

- Em geral, tem-se em um programa de computação gráfica, inúmeros tipos de figuras como círculos, arestas, polígonos convexos, esferas e poliedros.
- Para cada par te formas, existem algoritmos de detecção.
- O mais simples é o círculo-círculo que verifica, se a distância entre os centros é menor ou igual que a soma dos raios. Em caso afirmativo, há contato, caso contrário, não.

Teorema do eixo separador

- Para cada par de formas, existe um número infinito de eixos que podem ser o eixo de separação.
- Deve se escolher adequadamente esse eixo já que essa é uma etapa crucial da implementação do algoritmo de separação de eixos.
- Para polígonos convexos, pode-se utilizar a normal de uma aresta como potencial eixo separador.
- Para cada eixo do polígono, necessita-se apenas verificar se os vértices do outro polígono se encontram na frente dessa aresta

Teorema do eixo separador



Teorema do eixo separador

- Se, para uma determinada aresta, todos os vértices do outro polígono estiverem na frente dessa aresta, não há intersecção entre os polígonos.
- Seja uma aresta com vértices a e b e um vértice v que pertença ao outro polígono. Se $(v - a) \cdot n$ é maior que zero, então o vértice está na frente da aresta.

Teorema do eixo separador

- Para poliedros, pode-se utilizar a face normal e também o produto vetorial de todas as combinações de arestas para cada forma.
- Apesar desse trabalho parecer extenso, utiliza-se como um chute inicial de eixo o eixo encontrado numa etapa anterior (que deve ser devidamente guardada para esse fim).
- Essa abordagem funciona bem se não há grandes movimentações das formas entre os instantes de análise.

O algoritmo Gilbert-Johnson-Keerthi

- No caso de contato entre objetos, muitas vezes deseja-se não só saber se eles estão colidindo ou se interceptando, mas também se eles estão próximos.
- Essas informações requerem o cálculo da distância entre as formas.
- O algoritmo Gilbert-Johnson-Keerthi (GJK) calcula a distância entre duas formas convexas e também os pontos mais próximo entre os objetos.

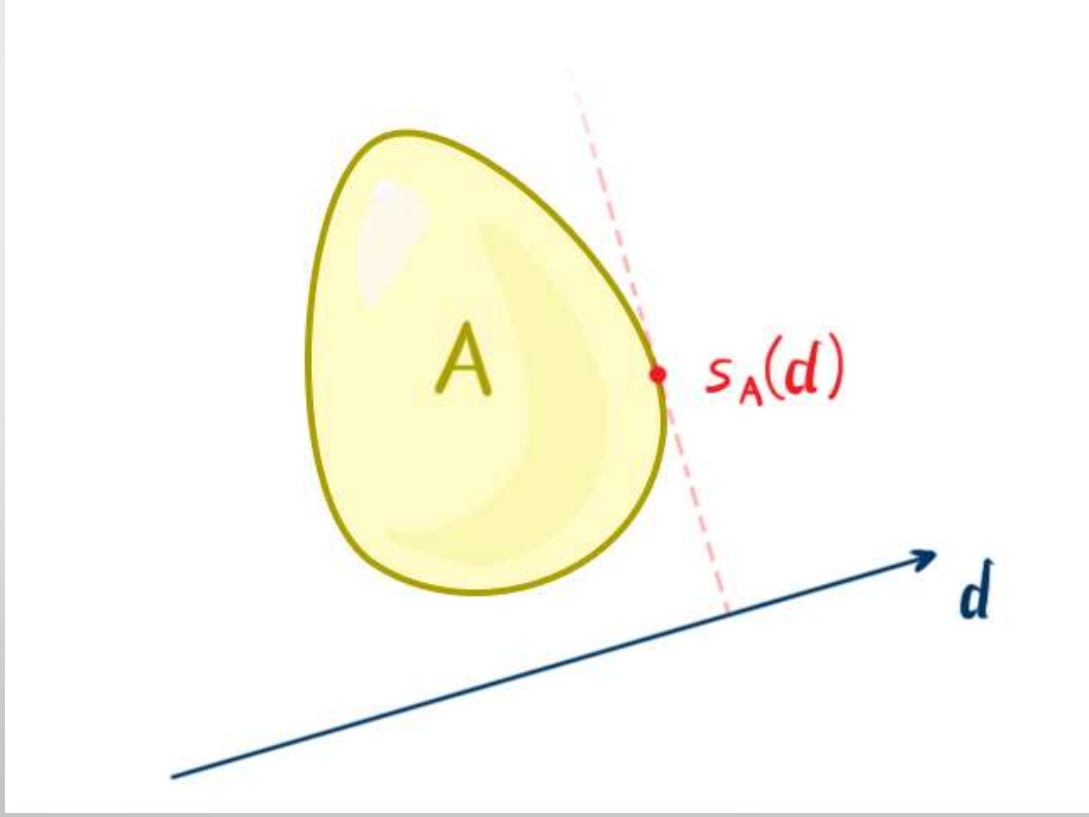
O algoritmo Gilbert-Johnson-Keerthi

- Esse é um algoritmo elegante que trabalha com uma representação implícita de figuras convexas através de:
  - Funções de suporte.
  - Somas de Minkowski.
  - Simplexes.

Funções de Suporte

- A função de suporte $s_A(\boldsymbol{d})$ retorna o ponto da fronteira da figura A que tem a maior projeção no vetor $\boldsymbol{d}$.
- Matematicamente, é o maior produto escalar com $\boldsymbol{d}$.
- Esse ponto é dito um ponto de suporte e a operação para obtê-lo é chamada de mapeamento de suporte.
- Geometricamente, esse ponto é o ponto mais afastado da figura na direção de $\boldsymbol{d}$.

Funções de Suporte



Funções de Suporte

- Encontrar o ponto de suporte de um polígono é relativamente simples.
- Para um vetor $\boldsymbol{d}$, o ponto de suporte é encontrado varrendo os vértices do polígono e encontrando aquele tal que o produto escalar com $\boldsymbol{d}$ é máximo.
- Essa formulação mostra sua real capacidade pois torna simples trabalhar com figuras como cones, cilindros e elipses, entre outras.

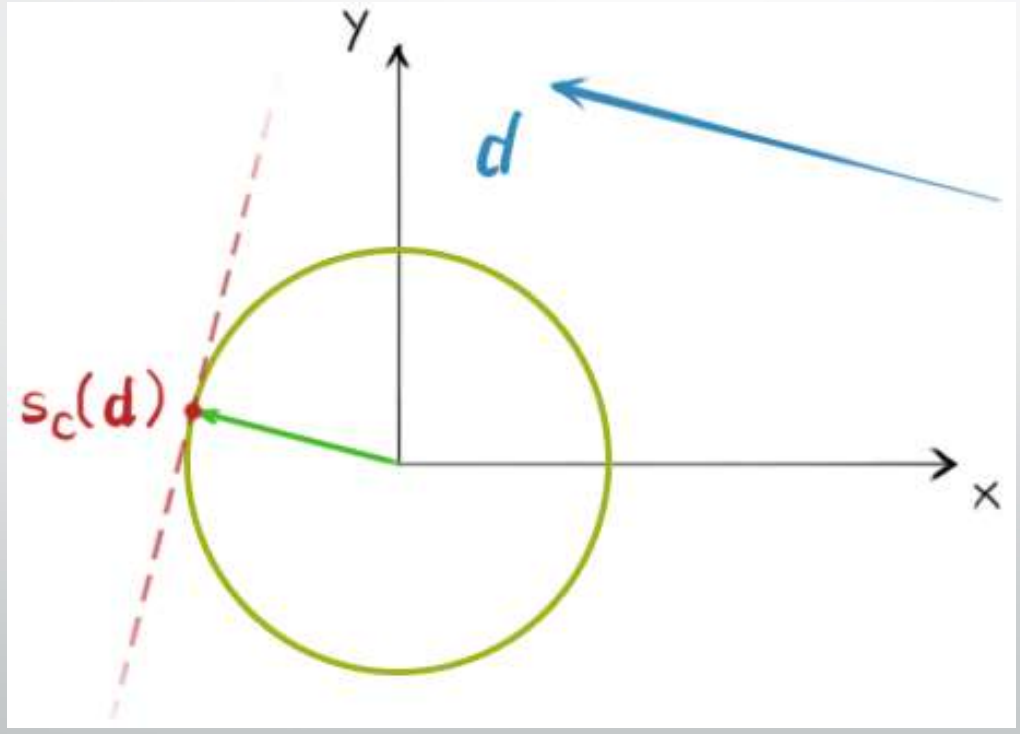
Funções de Suporte

- Computar a distância entre objetos dessas formas é difícil e sem um algoritmo como o GJK, seria necessário discretizar as superfícies tornando-as polígonos ou poliedros.
- Isso pode levar a problemas maiores já que a superfícies de um poliedro não é uma superfície suave.

Funções de Suporte

- Outro possível problema é, por exemplo, quando se trata de um movimento.
- Se uma esfera discretizada rola sobre um plano, esse movimento não será tão liso quando se espera.
- Para obter-se um ponto de suporte de uma esfera centrada na origem, normaliza-se $\boldsymbol{d}$ e se multiplica pelo raio da esfera.

Funções de Suporte



Funções de Suporte

- Os objetos que são tratados em programas computacionais são rotacionados e deslocados no espaço.
- Sendo assim, deve-se calcular os pontos de suporte para uma forma transladada:

$$s_{T(A)}(\boldsymbol{d}) = \boldsymbol{T}(s_A(\boldsymbol{R}^{\boldsymbol{T}}\boldsymbol{d}))$$

- onde $\boldsymbol{R}$ é a matrix de rotação e $\boldsymbol{T}$ é a translação do corpo.

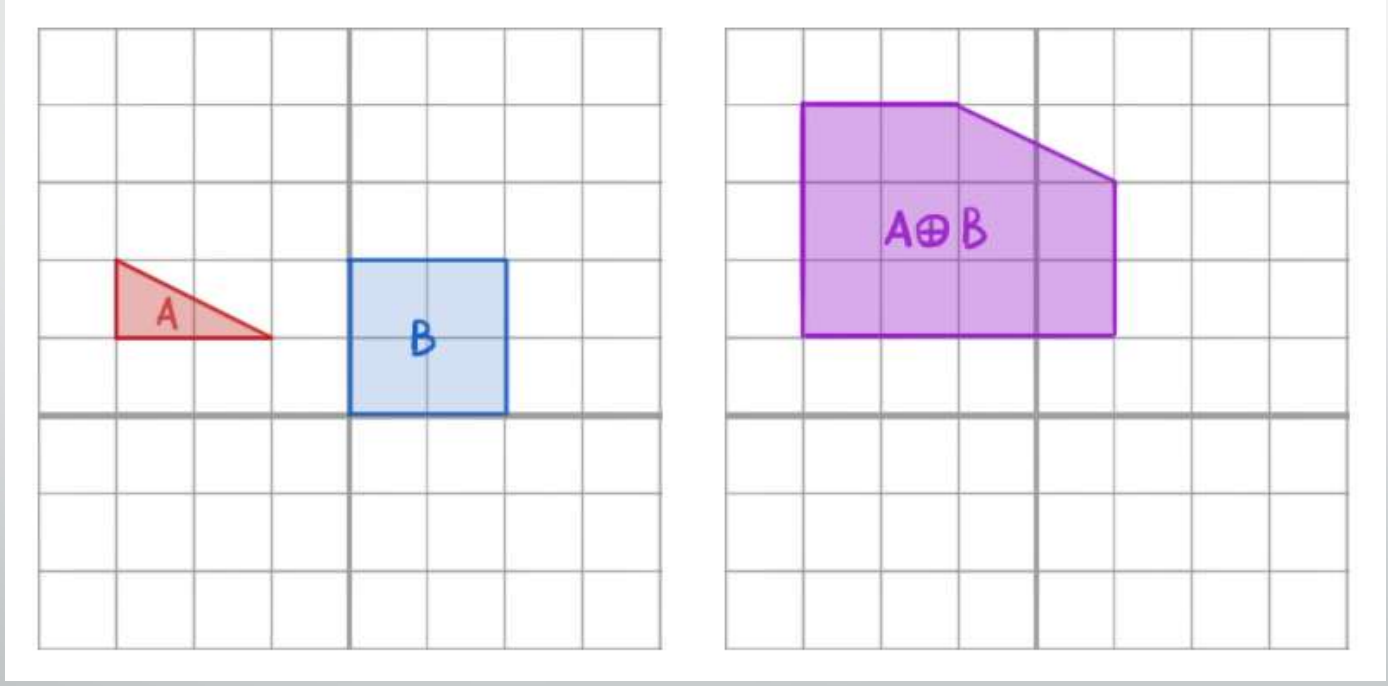
Soma de Minkowski

- A soma de Minkowski de duas formas A e B é definida como:

$$A \oplus B = \{\boldsymbol{a} + \boldsymbol{b} : \boldsymbol{a} \in A, \boldsymbol{b} \in B\}$$

- Isso significa que se soma todos os pontos contidos em A e B. O resultado é como se inflasse A com B.

Soma de Minkowski



Soma de Minkowski

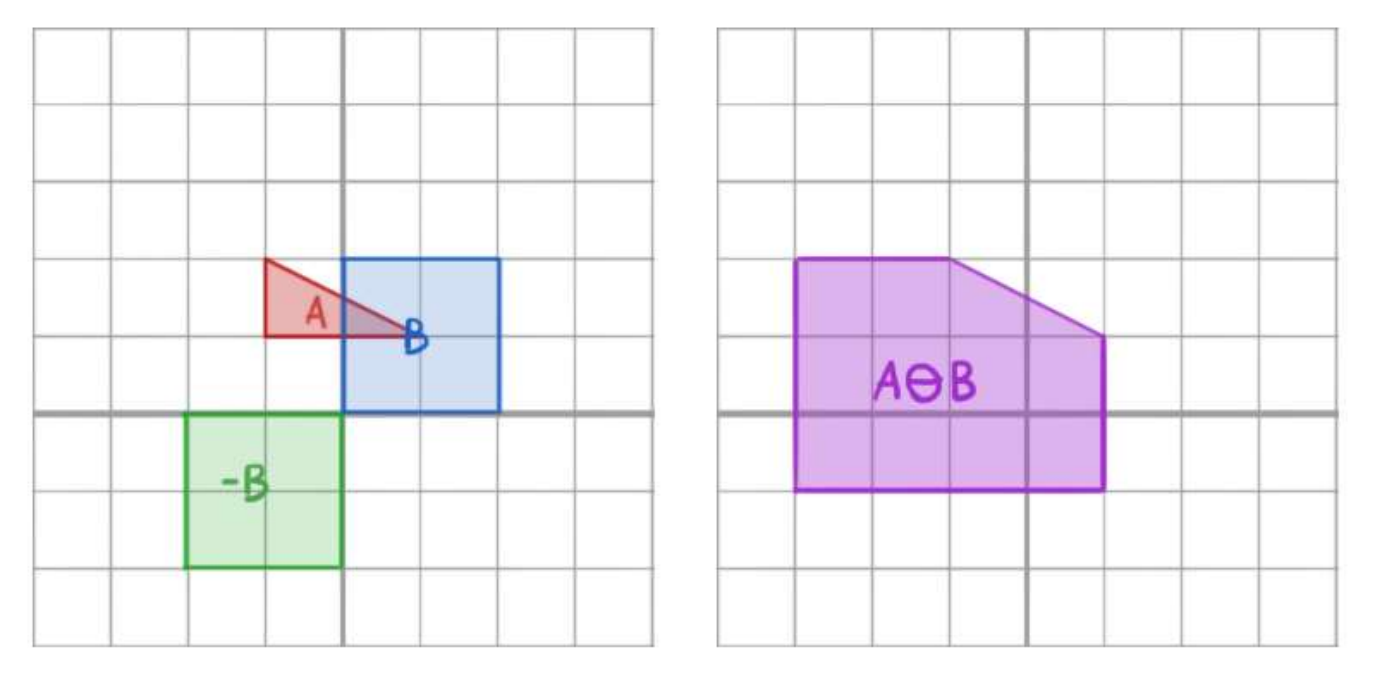
- De maneira similar, define-se a diferença de Minkowski como:

$$A \ominus B = \{\boldsymbol{a} - \boldsymbol{b} : \boldsymbol{a} \in A, \boldsymbol{b} \in B\}$$

- que pode ser escrita da soma de A com (-B):

$$A \ominus B = A \oplus (-B)$$

Soma de Minkowski



Soma de Minkowski

- Se A e B são convexos, $A \oplus B$ também é convexo.
- Uma propriedade da diferença de Minkowski é que se se ela contém a origem do espaço, as figuras se interceptam, como pode ser visto na figura anterior.
- Isso é explicado da seguinte forma: se as figuras se interceptam, elas têm pelo menos um ponto em comum e a diferença entre esses pontos resulta no vetor zero, que é a origem.

Soma de Minkowski

- Outra característica da diferença de Minkowski é que se a figura não contém a origem, a distância mínima entre a origem e a figura resultante da diferença é a própria distância entre as figuras.
- Sendo assim, a distância entre as figuras pode ser escrita como:

$$distancia(A, B) = \min\{\|\boldsymbol{a} - \boldsymbol{b}\| : \boldsymbol{a} \in A, \boldsymbol{b} \in B\}$$

Soma de Minkowski

- Ou seja, a distância entre A e B é o vetor de menor comprimento que vai de A até B.
- Esse vetor é contido na diferença entre A e B e é o que possui menor comprimento, consequentemente, o mais próximo da origem.
- Em geral não é simples construir a soma de Minkowski de duas figuras. Mas é possível expressá-la usando mapeamento de suportes:

$$s_{A \ominus B}(\boldsymbol{d}) = s_A(\boldsymbol{d}) - s_B(-\boldsymbol{d})$$

Simplexes

- O algoritmo GJK procura iterativamente pelo ponto da diferença de Minkowski que está mais próximo da origem.
- Para encontrar esse ponto, constrói-se uma série de *simplexes* que estão próximos à origem em cada iteração.
- Um *simplex*, ou mais especificamente, um k *−simplex* para um inteiro k, é um *convex hull* de $k + 1$ pontos independentes de um espaço k-dimensional.

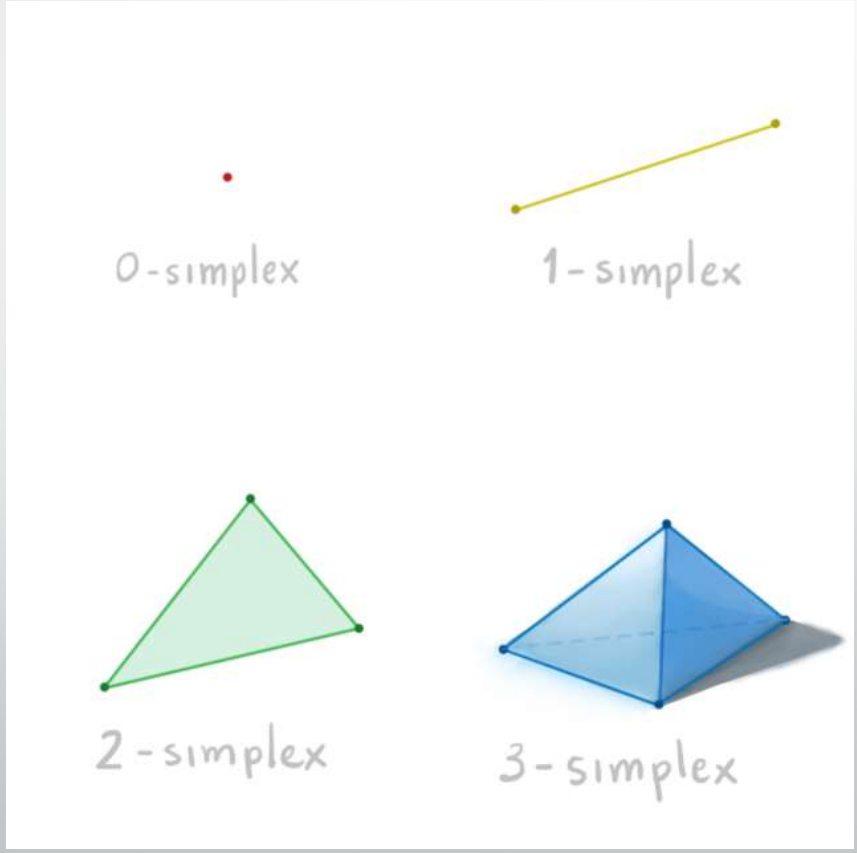
Simplexes

- Para dois pontos, eles não podem coincidir, para três pontos, eles não devem estar sobre a mesma linha, para quatro pontos, eles não podem estar no mesmo plano e assim sucessivamente.
- Assim, um 0-*simplex* é um ponto, 1-*simplex* é uma linha, 2-*simplex* é um triângulo e 3-*simplex* é um tetraedro.

Simplexes

- Para dois pontos, eles não podem coincidir, para três pontos, eles não devem estar sobre a mesma linha, para quatro pontos, eles não podem estar no mesmo plano e assim sucessivamente.
- Assim, um 0-*simplex* é um ponto, 1-*simplex* é uma linha, 2-*simplex* é um triângulo e 3-*simplex* é um tetraedro.
- Se remover-se um ponto de um *simplex*, decrementa-se a dimensionalidade do mesmo em 1 e se se adiciona um ponto, aumenta-se a dimensionalidade do *simplex* em um.

Simplexes



GJK em ação

- Para determinar a distância entre duas figuras A e B, inicia-se obtendo a diferença de Minkowski de A e B.
- Dessa forma, procura-se o ponto mais próximo da origem da figura resultante que, como foi dito anteriormente, é a distâncias entre as figuras originais.
- Escolhe-se um ponto v em $A \ominus B$, que será uma aproximação da distância.
- Define-se também um conjunto vazio de pontos W que conterá os pontos do *simplex* a ser testado.

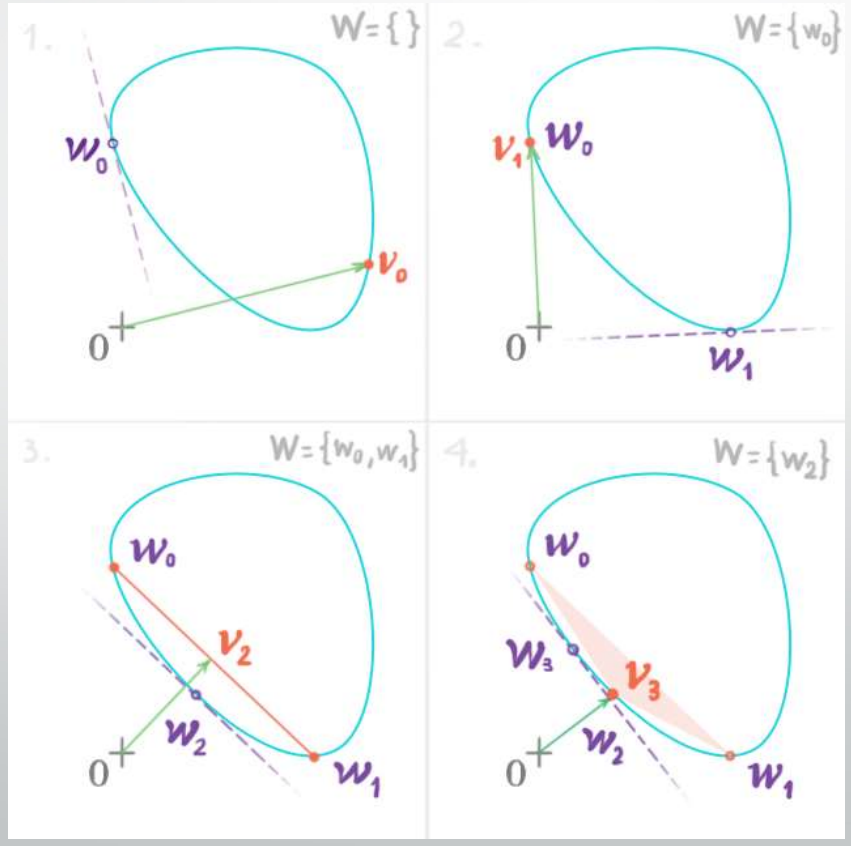
GJK em ação

- No próximo passo entra-se em um *loop.* Inicia-se tomando o ponto de suporte $\boldsymbol{w} = s_{A \ominus B}(-\boldsymbol{v})$, ou seja, o ponto de $A \ominus B$ cuja projeção em $\boldsymbol{v}$ é mais próxima da origem.
- Se a norma de $\boldsymbol{w}$ não for muito diferente da norma de $\boldsymbol{v}$ e o ângulo entre eles não mudar muito dentro de tolerâncias pré-estabelecidas, significa que o algoritmo convergiu e a distância é a $\|\boldsymbol{v}\|$.
- Caso contrário, adiciona-se o ponto $\boldsymbol{w}$ ao conjunto W.

GJK em ação

- Se o *convex hull* de W (ou seja, o *simplex*) contém a origem, então as figuras se interceptam, significando que o algoritmo terminou sua execução.
- Caso contrário, procura-se o no *simplex* o ponto mais próximo da origem e em seguida retornamos $\boldsymbol{v}$ ao seu valor inicial como a aproximação da distância.
- Por fim, remove-se quaisquer pontos de W que não contribuam para a aproximação.
- Repete-se esse processo até o algoritmo convergir.

GJK em ação



Determinando a penetração

- Para determinar-se a penetração entres as formas utiliza-se o *algoritmo de expansão dos polítopos (Expanding Polytope Algorithm).*
- Primeiramente deve-se definir o que é um polítopo.
- Ele é uma região contida no $\mathbb{R}^n$ que é resultante da intersecção de um conjunto de semi-espaços.
- Este conceito representa a generalização, para um número arbitrário de dimensões (finitas), dos conceitos de polígono e poliedro.

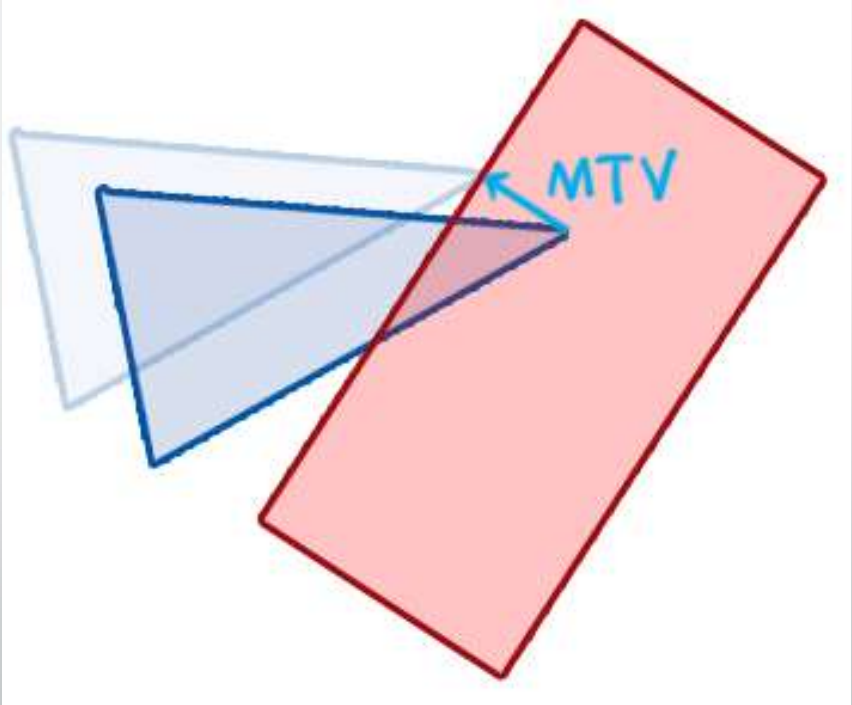
Algoritmo de expansão dos polítopos

- O algoritmo GJK gera um *simplex* que contém a origem, dentro da diferença $A \ominus B$.
- Até o presente momento, pode-se apenas dizer que as figuras interceptam.
- Porém, no processo de colisão (e detecção do contato) deseja-se saber quanto há de intersecção e quais são os pontos de contato, para que a colisão seja tratada de maneira realista.

Algoritmo de expansão dos polítopos

- O Algoritmo de expansão dos polítopos (*EPA* do inglês *Expanding Polytope Algorithm*) permite que se consiga tal informação partindo da informação obtida do algoritmo GJK: um *simplex* que contém a origem.
- A penetração é o comprimento do vetor de translação mínima (*MTV* ou do inglês *Minimum Translation Vector*), que é o vetor de menor comprimento que permite transladar uma das figuras de tal forma que elas se separem.

Algoritmo de expansão dos polítopos



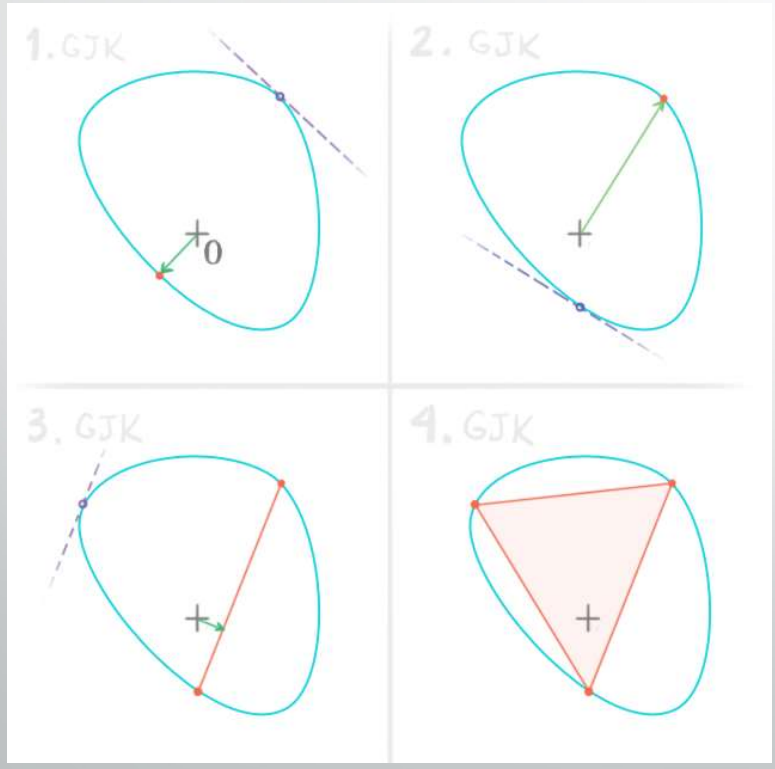
Algoritmo de expansão dos polítopos

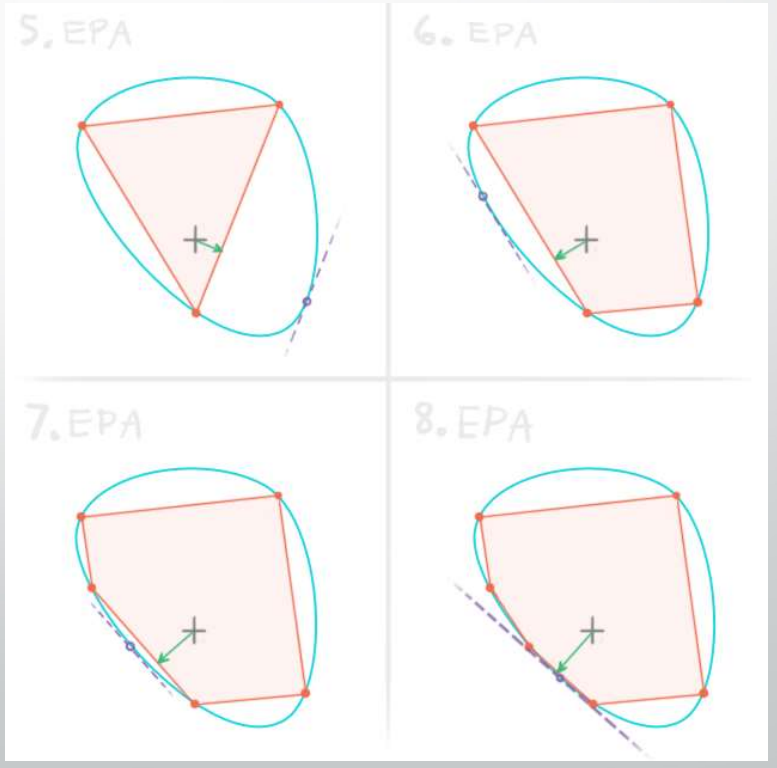
- Quando duas formas se interseccionam, a diferença de Minkowsi contém a origem e o ponto na fronteira da diferença que é o mais próximo da origem é o MTV.
- O algoritmo EPA encontra o ponto expandindo o *simplex* que o GJK retorna como polígono.
- Subdividindo sucessivamente os lados do polígono com novos vértices.

Algoritmo de expansão dos polítopos

- Primeiro, encontra-se o lado do simplex mais próximo da origem e se computa o ponto de suporte na diferença de Minkowski na direção da normal da aresta (ou seja, perpendicular a ela e apontando para fora do polígono).
- Em seguida, divide-se esse lado adicionando ponto de suporte a ele.
- Repete-se esses passos até a direção e comprimento do vetor não mudar muito.
- Uma vez que o algoritmo convergiu, tem-se o MTV e a profundidade de penetração.

Algoritmo de expansão dos polítopos





Detecção continuada de colisão

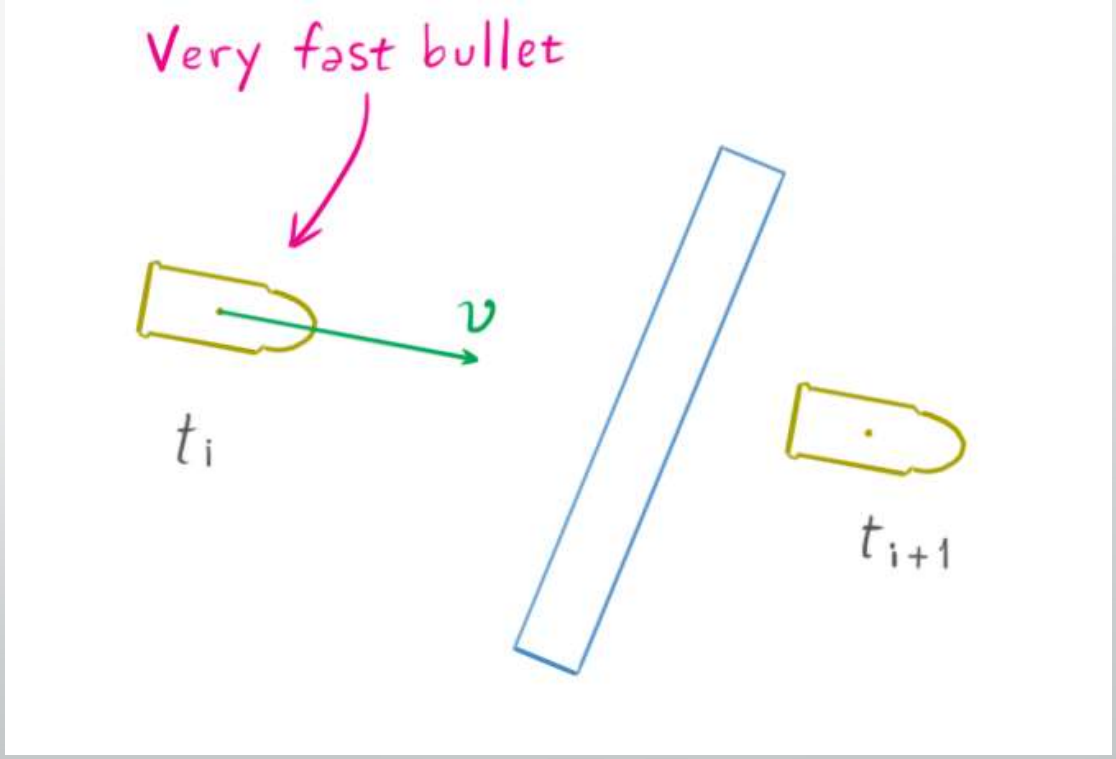
- Até o presente momento viu-se técnica que permitem detectar a colisão apenas em um momento específico da simulação.
- Essa técnica é chamada de detecção de colisão discreta, que ignora o que aconteceu entre o passo anterior e o passo atual.
- Por essa razão, algumas colisões podem não ser detectadas, especialmente para objetos que se movam em altas velocidades.
- Esse problema é conhecido como tunelamento (ou *tunneling* em inglês).

Detecção continuada de colisão

- Até o presente momento viu-se técnica que permitem detectar a colisão apenas em um momento específico da simulação.
- Essa técnica é chamada de detecção de colisão discreta, que ignora o que aconteceu entre o passo anterior e o passo atual.
- Por essa razão, algumas colisões podem não ser detectadas, especialmente para objetos que se movam em altas velocidades.
- Esse problema é conhecido como tunelamento (ou *tunneling* em inglês).

Detecção continuada de colisão

- A figura mostra dois instantes de tempo considerando uma bala e um anteparo. Caso se use a abordagem discreta, o contato não é registrado.



Detecção continuada de colisão

- A técnica detecção contínua procura encontrar dois corpos que colidam em um instante de tempo entre o anterior e o atual.
- Computa-se o tempo de impacto de tal forma que se possa voltar no tempo e ir para o instante do contato.

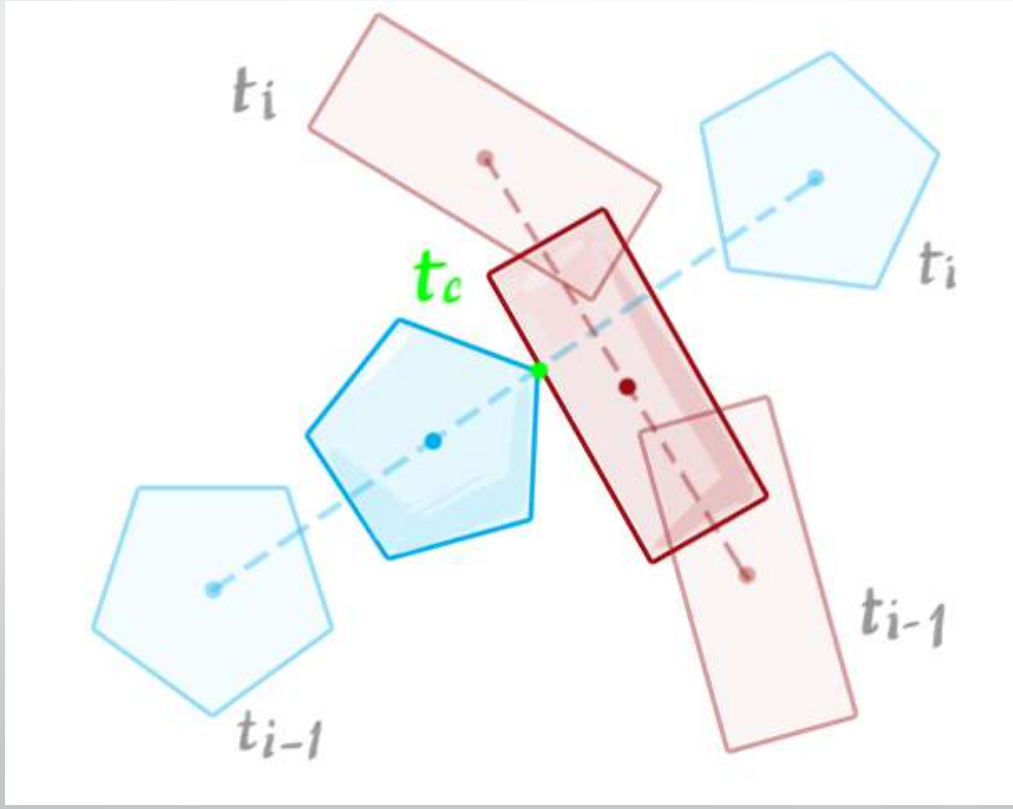
Detecção continuada de colisão

- O tempo de impacto (ou tempo de contato) t_C é o instante de tempo em que a distância entre os dois corpos é zero.
- Se for possível escrever uma função da distância entre os corpos no tempo, pode-se encontrar a menor raiz dessa função.
- Assim, o tempo de impacto computacional é um problema de determinar raízes de função.

Detecção continuada de colisão

- Para calcular o tempo de simulação, considera-se o estado (posição e orientação) do corpo no instante anterior ao tempo t_{i-1} e o tempo atual t_i.
- Para tornar as coisas simples, assume-se um movimento linear entre os passos de tempo.

Detecção continuada de colisão



Detecção continuada de colisão

- Pode-se simplificar o problema assumindo que as figuras são circulares.
- Seja dois círculos C_1 e C_2, com raios r_1 e r_2, com centros de massa $\boldsymbol{x_1}$ e $\boldsymbol{x_2}$ coincidentes com o centroide dos mesmos.
- Pode-se escrever a função que descreve a distância entre os corpos como:

$$d(t) = \|\boldsymbol{x_2} - \boldsymbol{x_1}\| - r_2 - r_1$$

Detecção continuada de colisão

- Essa expressão é uma interpolação linear de $\boldsymbol{x_1}(t_{i-1})$ até $\boldsymbol{x_1}(t_i)$.
- O mesmo pode ser feito para $\boldsymbol{x_2}$. Para esse intervalo pode-se escrever outra função distância:

$$d'(t) = \|\boldsymbol{x_2'} - \boldsymbol{x_1'}\| - r_2 - r_1$$

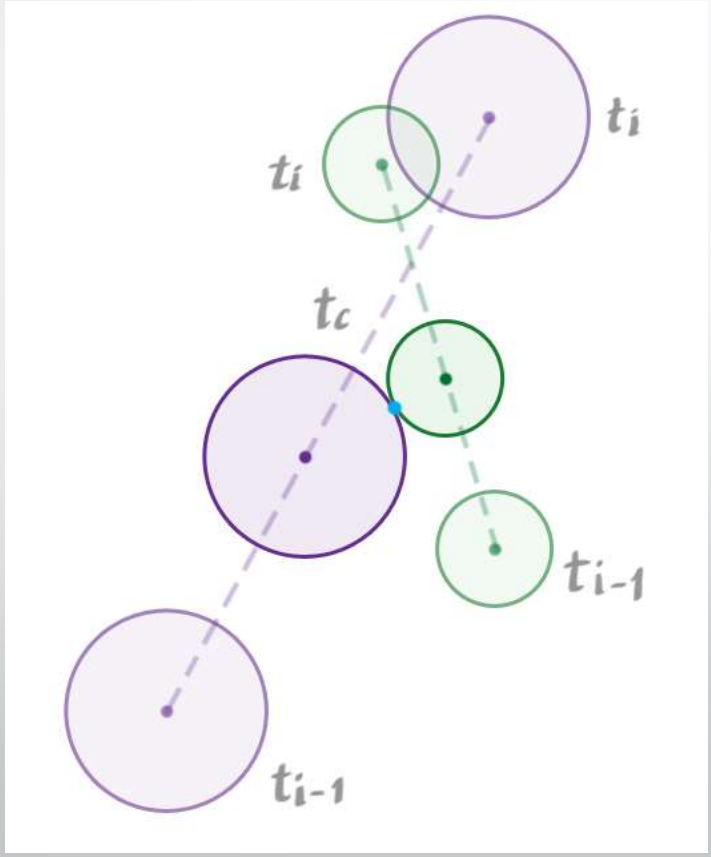
Detecção continuada de colisão

- Igualando essa expressão a zero, têm-se uma equação quadrática em t.
- As raízes podem ser encontradas diretamente através da fórmula quadrática.
- Se os círculos não se interseccionarem, a equação não terá solução.
- Se tiver, resultará em uma ou duas raízes.

Detecção continuada de colisão

- No caso de apenas uma raiz significa que o valor é o é o tempo de impacto.
- Se a equação tiver duas raízes, a menor é o tempo de impacto e o segundo valor é quando os círculos se separam.
- Note que o tempo de impacto aqui é um número de 0 a 1.
- Não é um tempo em segundos; somente um número que permite interpolar o estado dos corpos até a posição precisa na qual a colisão ocorre.

Detecção continuada de colisão



Detecção de colisão contínua para não círculos

- Escrever a função distância para outras formas é uma tarefa complicada, porque a distância depende da sua orientação.
- Em geral, utiliza-se algoritmos olham quadro a quadro o contato aproximando mais e mais as formas até que se possa considerar que as formas estão em contato.
- O algoritmo de avanço conservativo move os corpos para frente (e também os rotaciona) iterativamente.
- Em cada iteração, calcula um limite superior para o deslocamento.

Detecção de colisão contínua para não círculos

- O algoritmo computa os pontos mais próximos das figuras A e B, desenha um vetor de um ponto ao outro e projeta a velocidade nesse vetor para determinar um limite superior para o movimento.
- Ele garante que nenhum dos pontos do corpo moverá acima dessa projeção. O algoritmo avança os corpos nessa proporção e repete até que o valor da distância fique dentro de uma tolerância especificada.
- Pode ser que sejam necessárias várias iterações para que o algoritmo convirja, como por exemplo, quando a velocidade angular dos corpos é muito alta.

Resolvendo Colisões

- Uma vez que a colisão é detectada, é necessário mudar o movimento dos objetos de maneira realista, fazendo com que eles se choquem e movam em direções diferentes.
- Isso não está no escopo desse curso, mas encoraja-se o leitor a procurar métodos para esse cálculo.
- Cabe ainda lembrar que aqui se falou do contato entre superfícies rígidas.
- Quando os corpos são deformáveis, toda a teoria de contato e de mecânica dos sólidos pode ser empregada para calcular a forma final e o movimento desses corpos.