PSE Game Physics

Session (5)
Collision: Box-Box using separating axes theorem

Atanas Atanasov, Philipp Neumann, Martin Schreiber

27.05.2011

Outline

Separating axes

Theorem

Box-box intersections

Collisions: Box-Box in 2D

Extension to 3D

Towards implementation

Box-Box Intersections

- Box-Plane intersections quite complicated to compute using well-known techniques.
- When computing Box-Box intersections the same way, this results in a lot of complicated arithmetics.
  - “*A lot of*“: Many operations which we can avoid
  - “*Complicated*“: Error-prone (without using verification tools ;-)).
- **KISS**: Keep it simple and stupid!
- Therefore we utilize a more elegant and very efficient method.

Separating axes theorem (1/2)

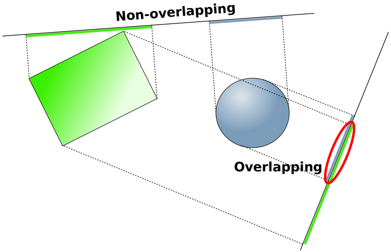


Figure: 2 examplary separating axes

Theorem

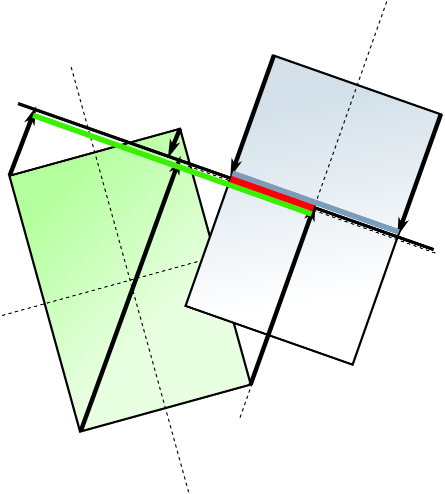
Separating Axes: For given convex objects, they do not intersect only iff there is a line on which the objects projections do not intersect.

This theorem is applicable only to **convex objects**!

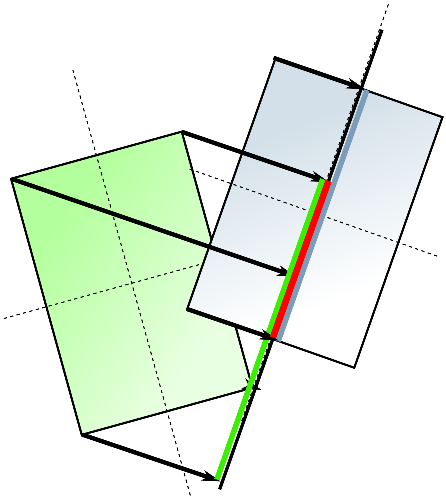
Separating axes theorem (2/2)

- We do not have to project our 2 objects on all existing axes.
- E. g. for 2D and 2 boxes, projecting on 4 axes parallel to the boxes-edges is sufficient.
- We can use this theorem to compute the point of intersections by doing simple projections on several axes.
- Next, we have a look at different projections for the 2D cases.

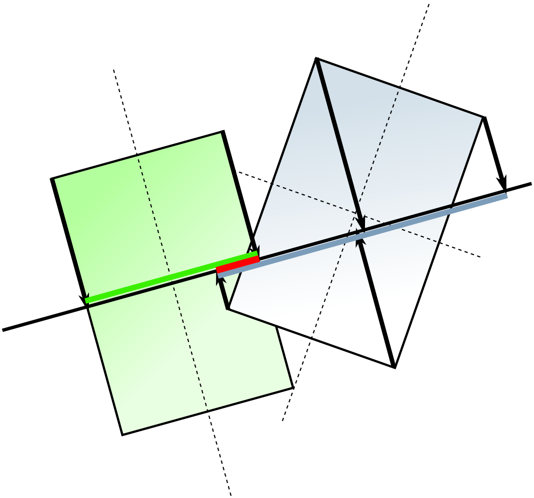
Collision: Box1, x-projection



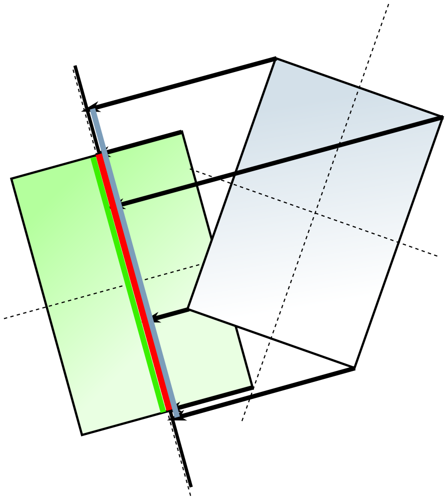
Collision: Box1, y-projection



Collision: Box2, x-projection



Collision: Box2, y-projection



Outline

Separating axes

Theorem

Box-box intersections

Collisions: Box-Box in 2D

Extension to 3D

Towards implementation

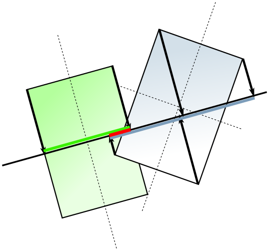
Collision

- There is **no separating axis** in our example and due to the theorem, **there has to be a collision**.
- **Question: Which “projected“ collision should we use?**

Collision

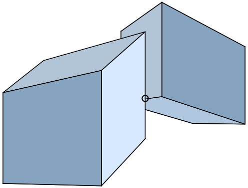
- There is **no separating axis** in our example and due to the theorem, **there has to be a collision**.
- **Question: Which "projected" collision should we use?**
- Answer: The collision with the **smallest interpenetration**!
- Computing interpenetration depth for one projection axis:
  - With x' as the first projected vertex, the projected interval is setup with $[x', x']$.
  - Then sucessively **extend this interval** by projecting the other projected vertices.
  - Do the same steps for the second object to create the projected interval of the second object.
  - Finally, check whether the interval of one object **overlaps** the interval of the other object.
  - In case that there is an interpenetration, **update the interpenetration depth** if the interpenetration is **smaller** than the previously detected.

Collision: Box2, x-projection



- Remember the data needed to resolve the interpenetration: Collision normal, Interpenetration depth, ...
- Having a look at the figure above, the **collision normal** is given by the **projection axis**.
- The **interpenetration depth** is directly given by the **size of the overlapping interval**
- Applying the separating axes theorem in 2D can be imagined as projecting each box into the other box's ”edge-space”.

Extension to 3D



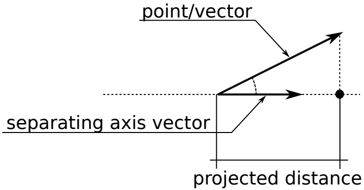
- In 3D, testing for separating axes which are aligned at the 6 box edges is **not sufficient**! ⇒ This would **not consider edge-edge** interpenetrations in 3D (see figure above).
- We also need to **test for possible non-interpenetrations of edge-edge**.
- To get an respective projection axis, we need an **axis perpendicular to both edges**! ⇒ Test for all **9 cross-product combinations** of principal axes of object 1 and object 2

Getting the separating axes vectors from the ”Edge-vectors”:

- The **3 edge-vectors** for a box in world-space are **equal to the principal axes** of the box
- The **3 principal axes** of the box are equal to the **first three model-matrix** colums!
  $\Rightarrow$ No need to do a matrix-vector multiplication

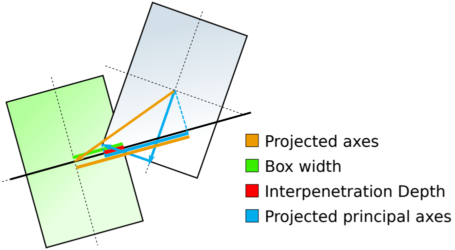
Projections

- Getting the projection distance on a normalized axis is equal to computing the dot product:



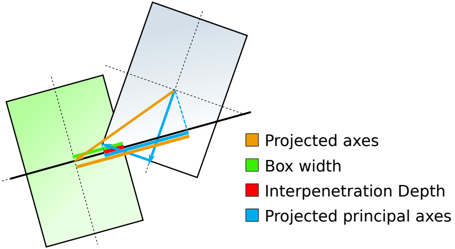
- Thus, just take $\vec{a} \cdot \vec{p}$ with $\vec{a}$ being one of the **normalized separating axes vectors** and $\vec{p}$ being a point to be projected to get the projected length.

Speeding things up (1/2)



- Remember: We are **only interested in the interval borders**, not every projection in particular.
- 3 projected distances are necessary to compute the interpenetration (see figure above):
  1. **Half-size of left box** in direction of separating axis
  2. Projected **distance of boxes midpoints**
  3. Projection of the vector from the middle of the right box to its vertex closest to the left box's center (next slide)

Speeding things up (2/2)



- To compute the “half-projected“ size of the right box, do the following computations for all 3 axes:
  1. **Project principal axis**.
  2. **Scale** projection by the box size related to the according direction.
  3. **Add the absolute value** to the interval size.
- Finally, we can compute the interpenetration distance

Collision points (1/3)

Question: How to determine collision points?

- Since very accurate collisions are not possible due to machine numbers.
- Thus we are allowed to **consider only Vertex-Face and Edge-Edge collisions** for our simplified engine.
- All other collisions (e. g. Vertex-Vertex, Vertex-Edge, ...) are almost impossible and would immediately occur within the next timestep.

Collision points (2/3)

Vertex-Face:

- When using a **box-principal axis as a separating axis** during detection of the smallest interpenetration, vertex-face collisions are considered only!
- The **collision normal** is almost directly **given by the separating axis**!!!
- You only have to consider the **direction of the collision normal** by **taking the centers of the boxes** into account since there are 2 possibilities for a single separating axis!

...

Collision points (2/3)

...

- The vertex which is the closest one to the object's face is taken as one of the collision points:
  - Use a projection similar to the one used in the section 'Speeding things up' by adding the 3 boxes half-axis vectors aiming in the direction of the collision.
  - Choose the right vector of the two available vectors representing one axis:
    The one is used which angle to the collision normal of the other object is larger.
- The **interpenetration depth** is given directly by the **overlapping distance** of the projections on the separating axis.
- Finally, the collision point on the face can be computed using the vertex position, the interpenetration depth and the normal.

Collision points (3/3)

Edge-Edge:

- When using non-box-principal axes as a separating axes (those created by x-product), edge-edge collisions are considered only.
- The interpenetrating edges have to be determined:
  - Since **one principal axis was used for the creation of the separating axis**, this principal axis is reused as the **direction vector of the edge**.
  - Then the **other two principal axes can be used to determine a single point on the colliding edge**.
  - By doing the previously described operations for both objects, both colliding edges are described each by one point on the middle of the edge and a direction vector.
  - Using a 'closest points on 2 lines' algorithm (e. g. `http://paulbourke.net/geometry/lineline3d/`), all necessary collision data can be computed.