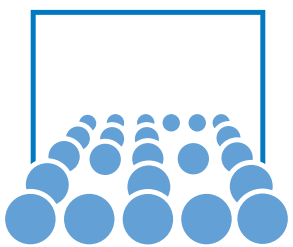
PSE Game Physics

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

Oliver Meister, Roland Wittmann

16.05.2014



Outline

Box-Box Intersections

- Box-Plane intersections are quite complicated to compute with the techniques we know so far.
- When computing Box-Box intersections the same way, this results in **a lot of** and **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)

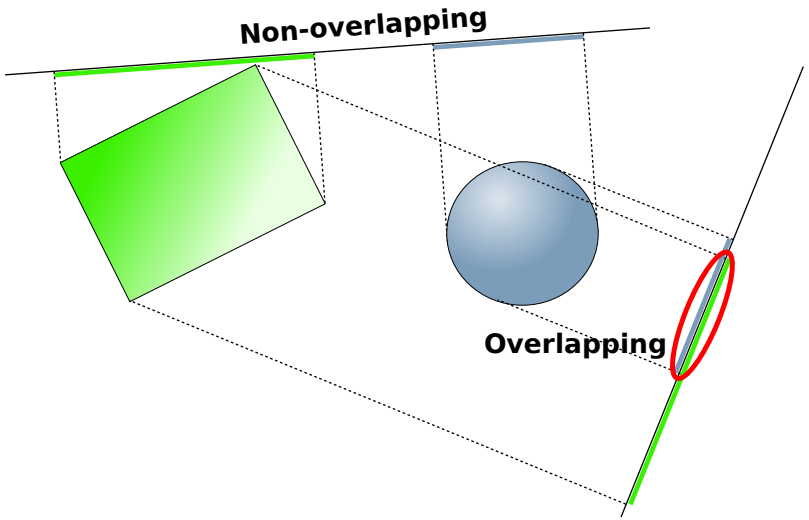


Figure: 2 examplary separating axes

Theorem

Separating Axes: Two convex objects do not intersect if and only if there is a line, which the objects projections do not intersect on.

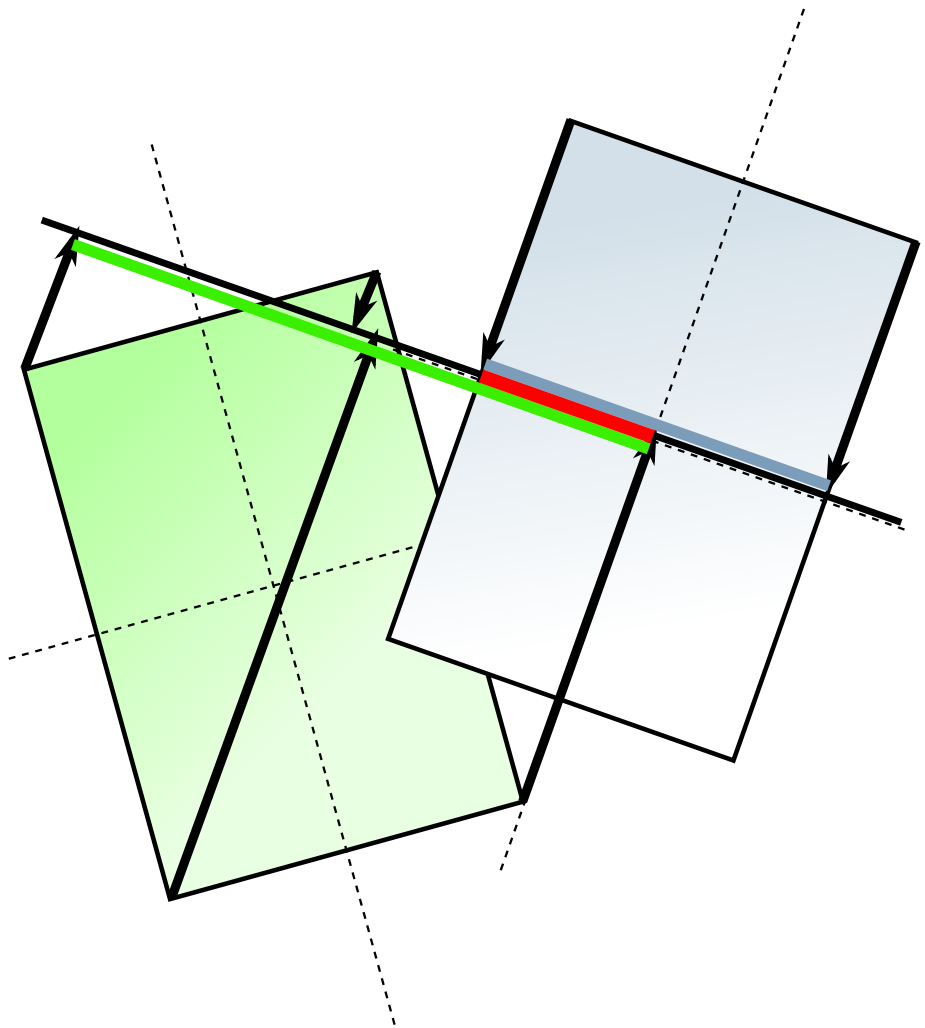
This theorem is applicable only to **convex objects**!
For all other objects, only the reverse implication $\Leftarrow$ holds.

Separating axes theorem (2/2)

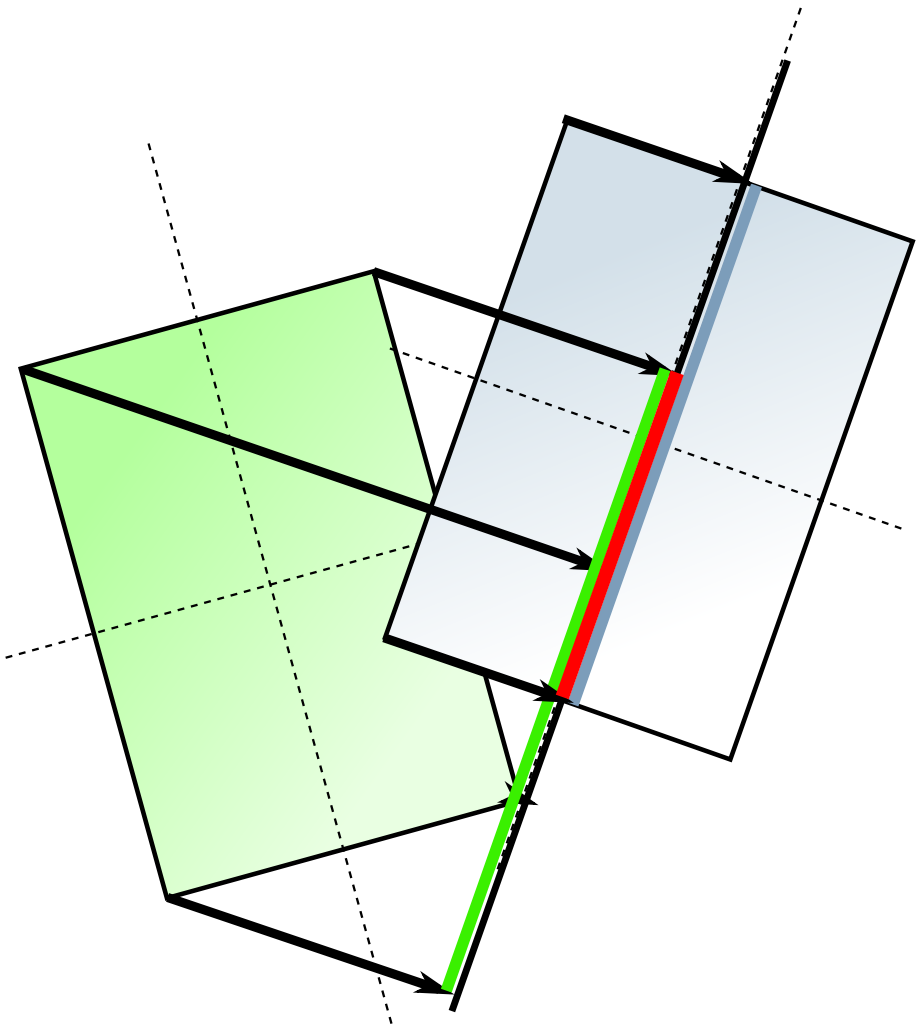
- We do not have to project our 2 boxes on all (infinite number) existing axes.
- E. g. for 2D and 2 boxes, **projection on 4 axes parallel to the box edges** is sufficient.
- We can use this theorem to compute the point of intersections by doing simple projections on several axes.
- Next, we have a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

Question: In the general case, should we always use axes parallel to the object surface/edges?

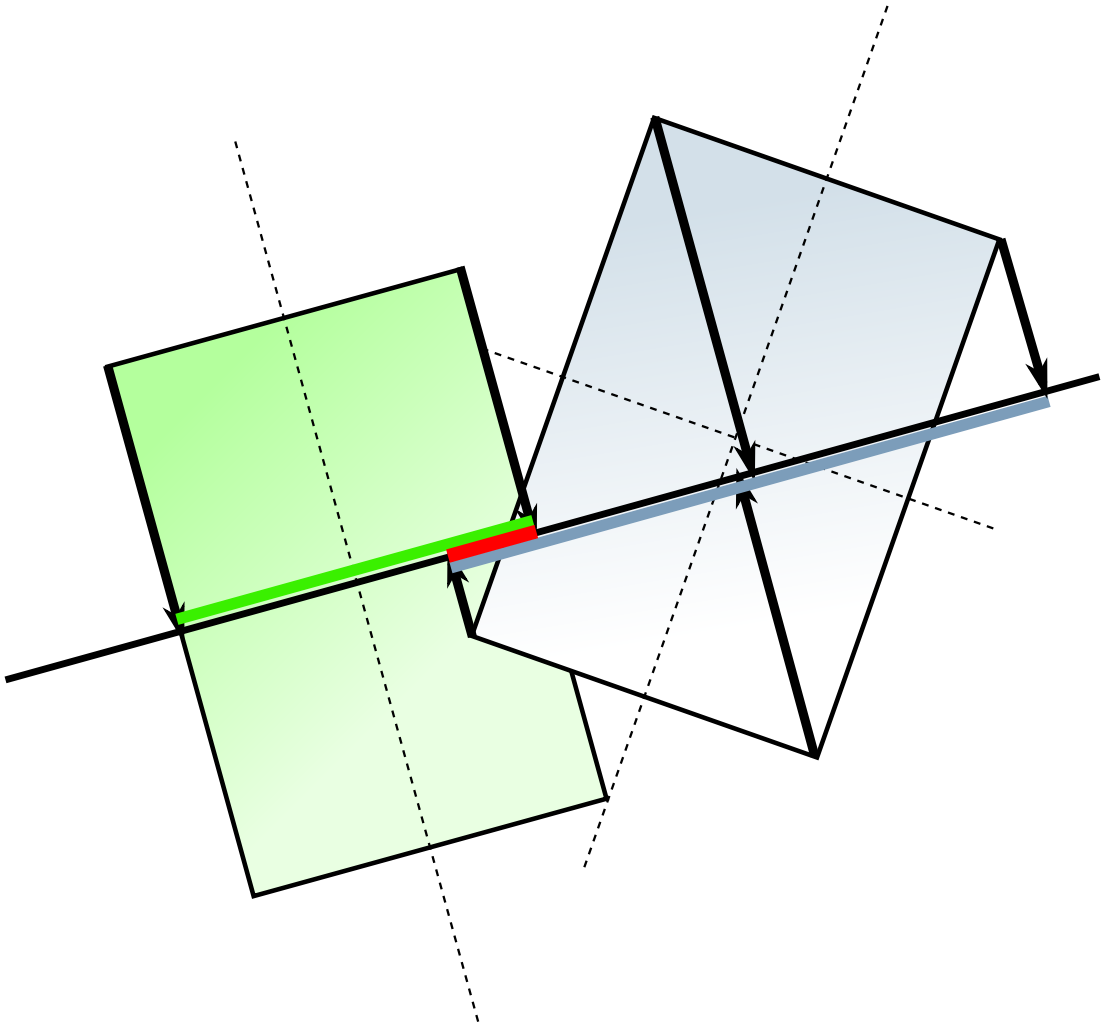
Collision: Box1, x-projection



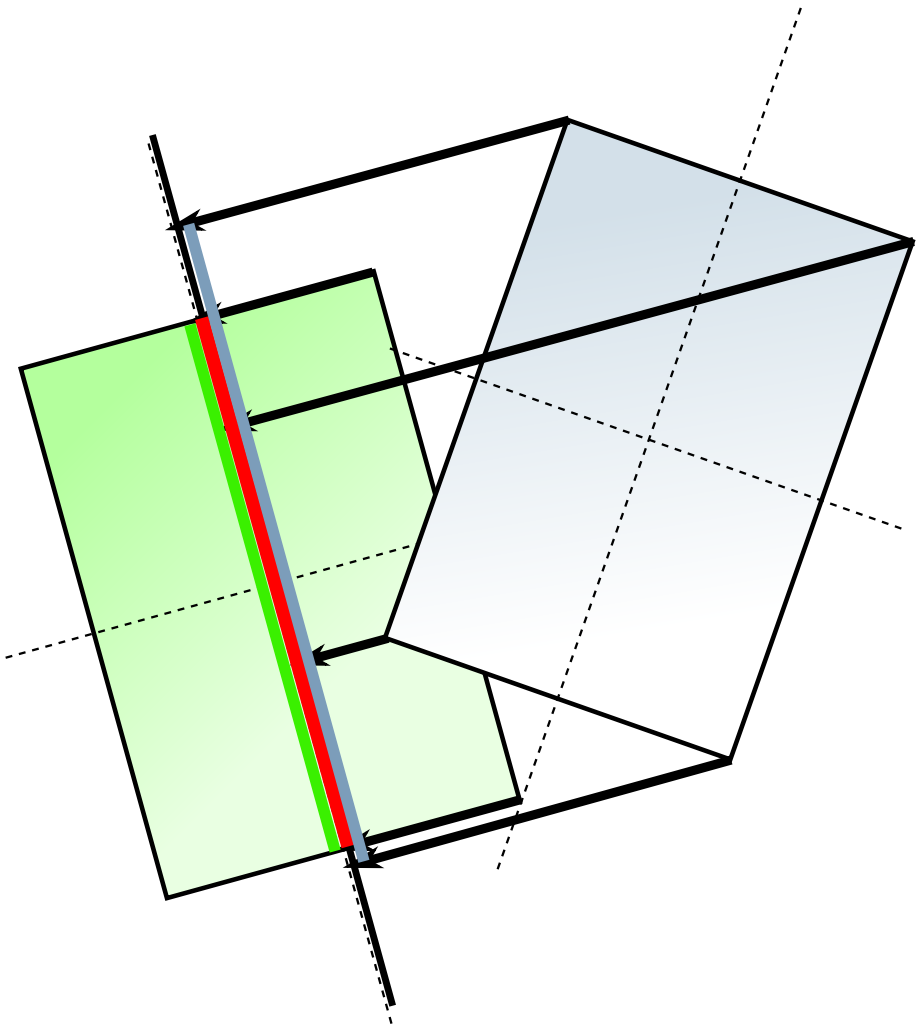
Collision: Box1, y-projection



Collision: Box2, x-projection



Collision: Box2, y-projection



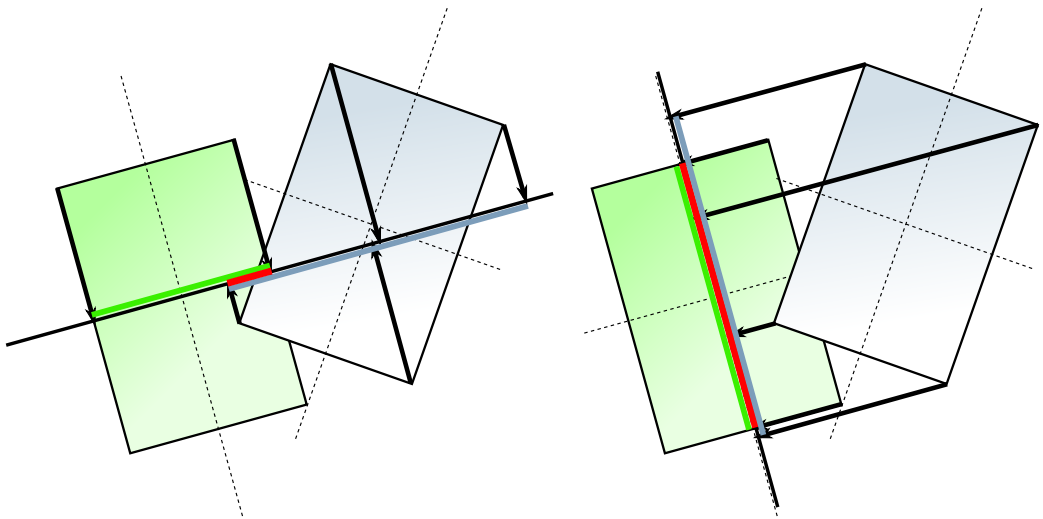
Outline

Separating axes
- Theorem

Box-box intersections
- Collisions: Box-Box in 2D
- Extension to 3D

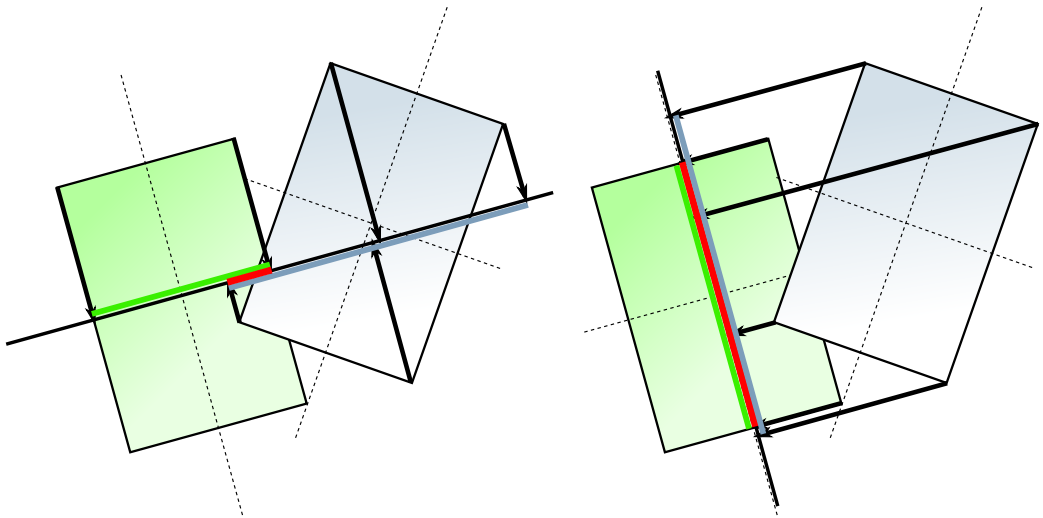
Collision

- 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?**



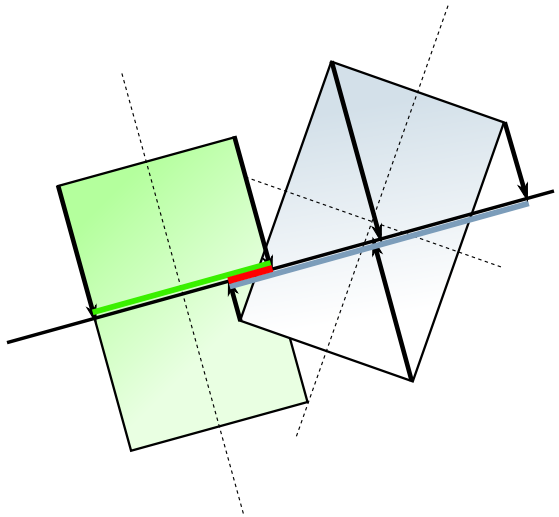
Collision

- 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?**



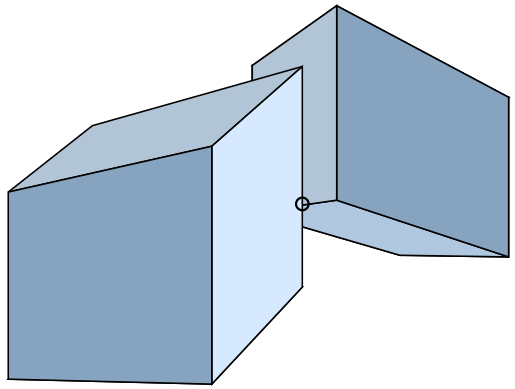
- Answer: The collision with the **smallest interpenetration**!

Collision normal & interpenetration depth



- 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



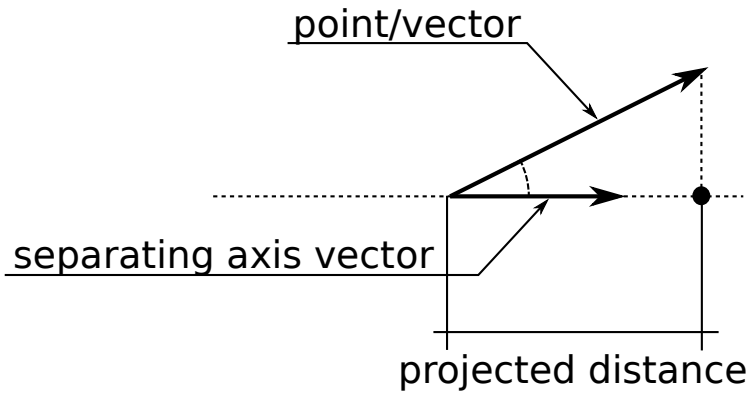
- In 3D, testing for separating axes which are parallel to the 6 (of 24) box edges is **not sufficient**! $\Rightarrow$ This would **not consider edge-edge** interpenetrations in 3D (see Figure above).
- We also need to **test for possible edge-edge non-interpenetrations**.
- To get a respective projection axis, we need an **axis perpendicular to both edges**! $\Rightarrow$ 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 (of 12) edges** of 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-world-matrix** colums!
  $\Rightarrow$ No need to do a matrix-vector multiplication to project basis-vectors to world-space.

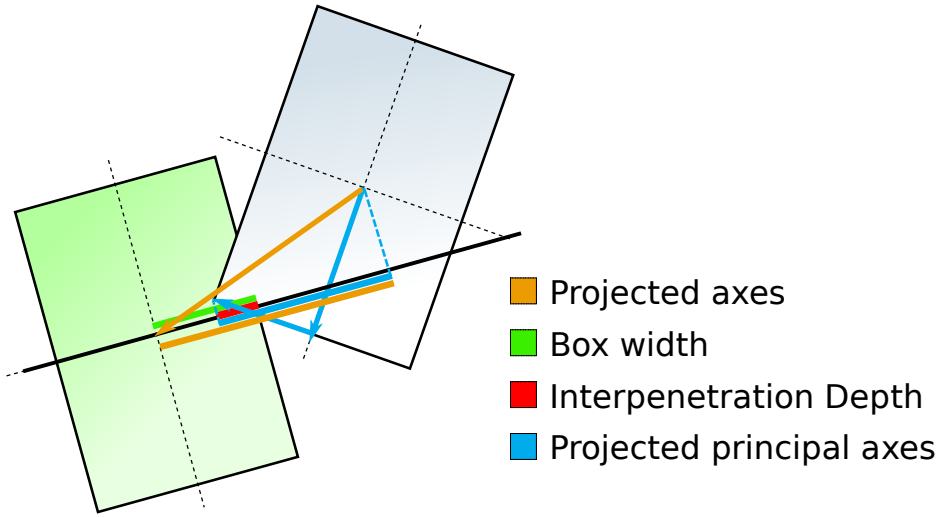
Projections

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



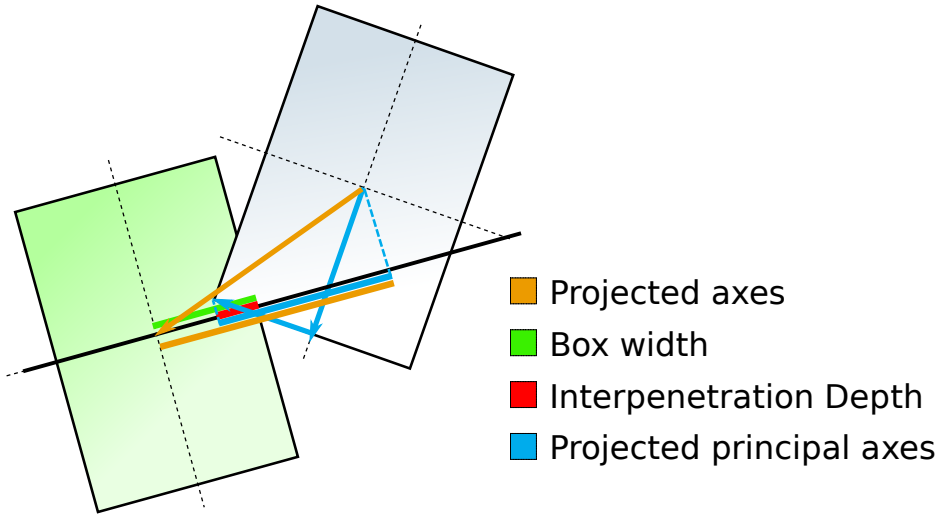
- Thus, just take $\vec{a} \cdot \vec{v}$ with $\vec{a}$ being one of the **normalized separating axes** and $\vec{v}$ being a vector to be projected to get the projected length.

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 for 2D):
  1. **Half-size of left box** in direction of separating axis
  2. Projected **distance of box 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 of the right box:
  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?

- Very accurate collisions are not possible due to limitation of 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:

- Vertex-face collisions require **principal axes as separating axes**.
- 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 closest to the object's face is taken as one of the collision points:
  - Use a projection similar to section 'Speeding things up' by **adding the 3 box half-axes aimed in the direction of the other box center**.
  - On each axis, we **walk either forward or backward**: The **larger angle with the collision normal** of the other object decides.
- 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 **collision normal**.

Collision points (3/3)

Edge-Edge:

- Edge-edge collisions require axes created by cross products, not principal axes.
- The interpenetrating edges have to be determined for each box:
  - 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 the colliding edge**.
  - By doing the previously described operations for both objects, both colliding edges are described by one point in the edge's midpoint and a direction vector.
  - Using a 'closest points on 2 lines' algorithm, both collision points can be computed.