Collision detection & response

Axially aligned bounding box (AABB)

Closest Point in an AABB

- https://gamemath.com/book/geomtests.html#closest_point_aabb

```
if (x < minX) {
    x = minX;
} else if (x > maxX) {
    x = maxX;
}

if (y < minY) {
    y = minY;
} else if (y > maxY) {
    y = maxY;
}

if (z < minZ) {
    z = minZ;
} else if (z > maxZ) {
    z = maxZ;
}
```

```
# Closest point in bbox
# Returns x,y if inside bbox
def cpbbox(
  x,y,
  xmin,ymin,xmax,ymax
 )
 if x<xmin
  x=xmin
 elsif x>=xmax
```

```
  x=xmax-1
 end
 if y<ymin
  y=ymin
 elsif y>=ymax
  y=ymax-1
 end
 return x,y
end
```

> This is done by “pushing” into along each axis in turn. Notice that if the point is already inside the box, this code returns the original point.

Intersection of a Sphere and AABB

- https://gamemath.com/book/geomtests.html#intersection_sphere_aabb

```
# Stationary circle with stationary
#  bbox collision
def circbbox(
  x,y,r,
  xmin,ymin,xmax,ymax
)
 cx,cy=cpbbox(x,y,xmin,ymin,xmax,ymax)
 (x-cx)**2+(y-cy)**2<=r**2
end
```

> ... we first find the point on the box that is closest to the center of the sphere by using the... Closest Point in an AABB... We compute the distance from this point to the center of the sphere and compare this distance with the radius. (Actually, in practice we compare the distance squared against the radius squared to avoid the square root in the distance computation.) If the distance is smaller than the radius, then the sphere intersects the AABB.

Intersection of Two AABBs

- https://gamemath.com/book/geomtests.html#intersection_two_aabbs

```
bool aabbsOverlap(const AABB3 &a, const AABB3 &b) {

    // Check for a separating axis.
    if (a.min.x >= b.max.x) return false;
    if (a.max.x <= b.min.x) return false;
    if (a.min.y >= b.max.y) return false;
    if (a.max.y <= b.min.y) return false;
    if (a.min.z >= b.max.z) return false;
    if (a.max.z <= b.min.z) return false;

    // Overlap on all three axes, so their
    // intersection must be non-empty
    return true;
}
```

```
# Stationary bbox and stationary
#  bbox collision
def bboxbbox(
  aminx,aminy,amaxx,amaxy,
  bminx,bminy,bmaxx,bmaxy
 )
 not (
  aminx>=bmaxx||amaxx<=bminx||
  aminy>=bmaxy||amaxy<=bminy
 )
end
```

Test if left is on the right or right on the left or top below bottom or bottom above top (same front and back for 3D) - not overlapping.

There is a dynamic test described which tells interval of time when the two boxes overlap when the collision starts given velocity vector.

Dynamic intersection of Two AABBs

- https://gamemath.com/book/geomtests.html#intersection_two_aabbs

> ... determine the first point in time when the boxes have overlap on all dimensions simultaneously ...

One static and one moving

```
# Overlap of stationary segmetn with
#  dynamic segment moving by v
# Returns [time enter, time leave]
# time enter:
#□□[0.0,1.0> - s and m will overlap
#□□□next frame
#  <0.0 - s is inside m
#□□-Inf□- s is inside m and v is zero
#  >1.0 - s will overlap with m in
#□□ time enter frames
#□□+Inf - s and m will never overlap
#   as v is zero
def segseg_dyn(
□□smin,smax,
□□mmin,mmax,
□□v
□)
□if v!=0
□□te=(smin-mmax).to_f/v
□□tl=(smax-mmin).to_f/v
□□te,tl=tl,te if te>tl
□□return te,tl
□else
□□# Not moving - static check
□□if not (smin>=mmax||smax<=mmin)
□□□# Always overlaps
□□□return -Float::INFINITY,Float::INFINITY
□□else
□□□# Never overlaps
□□□return Float::INFINITY,Float::INFINITY
□□end
□end
end
```

```
# Stationary with moving bbox collision
# Returns same value as segseg_dyn but
#  for 2 dimensions both overlapping
def bboxbbox_dyn_iv(
  sminx,sminy,smaxx,smaxy,
  mminx,mminy,mmaxx,mmaxy,
  vx,vy
 )
 tex,tlx=segseg_dyn(
  sminx,smaxx,
  mminx,mmaxx,
  vx
 )
 tey,tly=segseg_dyn(
  sminy,smaxy,
  mminy,mmaxy,
  vy
 )
 te=[tex,tey].max
 tl=[tlx,tly].min
 if te>=tl
  # No overlap - never collides
  return Float::INFINITY,Float::INFINITY
 else
  return te,tl
 end
end

# Stationary with moving bbox collision
# Returns multiplier value:
#  nil   - won't collide next frame
#  [0,1> - will collide next frame
#  <0    - collides already
#  -Inf  - collides already zero vx, vy
def bboxbbox_dyn(
  sminx,sminy,smaxx,smaxy,
  mminx,mminy,mmaxx,mmaxy,
  vx,vy
 )
 te,tl=bboxbbox_dyn_iv(
```

```
  sminx,sminy,smaxx,smaxy,
  mminx,mminy,mmaxx,mmaxy,
  vx,vy
 )
 if te>=0.0&&te<1.0
  # will collide in the next frame
  te
 elsif te<0.0&&tl>0.0
  # collides
  te
 else
  # won't collide
  nil
 end
end
```

Both moving

```
# Moving with moving bbox collision
# Returns multiplier value:
#  nil   - won't collide next frame
#  [0,1> - will collide next frame
#  <0    - collides already
#  -Inf  - collides already zero vx, vy
def bboxbbox_dyn2(
  aminx,aminy,amaxx,amaxy,
  avx,avy,
  bminx,bminy,bmaxx,bmaxy,
  bvx,bvy
 )
 vx=bvx-avx
 vy=bvy-avy
 bboxbbox_dyn(
  aminx,aminy,amaxx,amaxy,
  bminx,bminy,bmaxx,bmaxy,
  vx,vy
 )
end
```

Revision #14
Created 2025-04-22 19:11:05 IST by hxd
Updated 2025-04-28 20:15:58 IST by hxd