**Physics update**

* Frame control
  + Is paused
    - Step one frame forward
  + If not pause update normally
    - Broad phase (Dynamic AABB tree quarry), for each object
      * Update bounding volume for each object
        + Rotate AABB defined in a object space for the mesh
        + Construct an AABB of that rotated AABB
      * Reinsert into the tree this AABB
        + Remove from the tree old AABB with preinitialized ID corresponding to this object
        + Add to the tree newly constructed AABB
      * Self query the tree to detect any possible collision between any bounding volumes (AABBs constructed earlier)
        + Query left side of the tree to the right side
        + Query left side of the tree
        + Query right side of the tree
      * Delete duplicate query results
      * Check with the collision table
        + If the objects not mapping to each other in the table then remove query result
    - Validate last frame contacts if they still relevant, for each contact keep contacts if:
      * The two objects in contact still penetrate
      * Coordinate of contact for object A did not change much
      * Coordinate of contact for object B dd not change much
    - Narrow phase, for each query result from broad phase
      * Construct support shape (OBB collider)
        + Keep reference to transform, global scale, rotation, local scale (if model is not from -0.5 to +0.5 space)
      * Run GJK algorithm to check collision between convex shapes (OBB in our case)
        + Calculate search direction

Global position of object A I position of object B

If this direction is zero vector then initialize to arbitrary (1,0,0)

* + - * + Compute supports for the objects (furthest point in the support shape defined earlier in the given direction)

Object A in the direction

Object B in the – direction

* + - * + Initialize simplex with this CSO (Configuration Space Obstacle) point
        + Run 100 times loop below

Identify in which Voronoi region is the origin in in respect to simplex

Calculate closest point using barycentric coordinates

Calculate new search direction from closest point to the origin

If the closest point that was found Is close enough to origin (within 0.001 distance)

Fill the simplex to tetrahedron

If the simplex is one point look for another point in principal axis directions and add it to simplex, continue filling simplex with 2 points

If simplex is two points expend simplex by searching for new point in principal axis directions and adding it to simplex, continue filling simplex with 3 points

If simplex is three points expend simplex by searching for a new point in the direction of the normal of a triangle that is formed by these 3 points and adding it to the simplex

Fix tetrahedron to be in CCW ordering

Terminate the GJK with the true result that the objects penetrate

Search for new support point for the object in the new search direction

If this new point is no further then the closet point then terminate GJK with false, there is no collision

Add the new point to simplex

* + - * Run EPA if GJK returns confirmation that OBBs penetrate
        + Start with the tetrahedron that GJK terminated

Init the 4 triangles (faces of the tetrahedron)

* + - * + Loop 50 times:

Find closest triangle to the origin

Calculate for next support point in front of the triangle away from the origin

If new point is now further away from the origin then this triangle

Store this point as point of contact

Store distance from origin to this new support point as depth

Store normal of the triangle as a normal of the collision

Terminate EPA

For each triangle if it is visible from the currently found support point

Add 3 edges from the support point to each point of the triangle

Remove the triangle

Initialize and add the triangles defined by new edges

* + - * Process new contact that EPA returns
        + Check if collision manifold already exist for these 2 objects (if they were already colliding last last frame) and if so

Loop through all contacts in manifold and see if this new contact is far enough away from the each contact, and if so

Build constraints for new contact

Normal constraint

2 friction constraints

Add contact to manifold

Keep only up to contacts in the manifold

Find deepest penetrating contact

Find furthest away contact point from deepest penetrating

Find furthest contact away from the line formed by the 2 contact points found earlier

Find fourth point away from the triangle formed by previously found points

Drop the rest of contacts, four is enough

* + - * + If this is new collision manifold

Add new collision manifold to the list of the collision manifold

Build constraints for new contact

Add contact to this new collision manifold

* + - Update velocity
      * Calculate gravity force
      * Calculate air drag force
      * Sum forces, divide them by mass, multiply by dt and add to last frame velocity
    - Solve constraints
      * For each manifold
        + Repeat 50 times (Projected Gauss-Seidel)

For each contact in manifold

Solve for normal constraint

Solve for both friction constraints

Update velocity with each constraint

* + - Update position
  + Position = position + velocity \* dt