CollisionPoint

+points: vec3[2]
+normal: vec3
+penetration: float
+objects: *PhysicsObject[2]

+CollisionPoint(in other:CollisionPoint)
+CollisionPoint(in point1:vec3,in point2:vec3,
in normal:vec3,in penetration:float)
+reverse(): CollisionPoint
+toString(): string

Contact

-localPoints: vec3[2]
-matContactToWorld: mat3
-matWorldToContact: mat3
-closingVelocityWorld: vec3
-closingVelocityContact: vec3
-ContactID: unsigned int
-restitutionCoef: float
-frictionCoef: float
-desiredDeltaVelocity: float
-timestamp: unsigned int

+Contact(in collisionPoint:CollisionPoint)
+setContactInfo(in point1:vec3,in point2:vec3,
in normal:vec3,in penetration:float,
in obj1:*PhysicsObject,in obj2:*PhysicsObject)
+toString(): string
-computeDerivedData(): void

ContactManifold

+obj1: *PhysicsObject
+obj2: *PhysicsObject
+contacts: vector<Contact*>
-timestamp: unsigned int

+getDeepestContact(): *Contact
+getFastestContact(): *Contact

SequentialImpulseContactResolver

-timestamp: unsigned int
+contacts: list<Contact>
+manifolds: list<ContactManifold>

-addContactToManifold(inout manifold:ContactManifold,
in contact:CollisionPoint): void
-findWorstContact(in manifold:ContactManifold): unsigned int
-solveContactManifold(in manifold:ContactManifold): void
-computeDesiredDeltaVelocity(contact:*Contact): float
-computeImpulse(contact:*Contact): vec3
-applyPositionUpdate(inout manifold:ContactManifold,
inout deepestContact:*Contact): void
-applyVelocityUpdate(inout manifold:ContactManifold,
inout fastestContact:*Contact): void
-updateContacts(in collisions:vector<CollisionPoint*>): void
+solve(in collisions:vector<CollisionPoint*>): void