1 Collision Model

The impulse over a single dt, which for this simulator is some fraction of the individual frame time is as follows.

$$\vec{\mathbf{J}}_A = \int_{t_0}^{t_1} \vec{\mathbf{F}}_A \, dt = \Delta p_a = |\vec{\mathbf{p}}_B| \cdot \begin{bmatrix} \cos(\theta_{p,b}) \\ \sin(\theta_{p,b}) \end{bmatrix} - |\vec{\mathbf{p}}_A| \cdot \begin{bmatrix} \cos(\theta_{p,a}) \\ \sin(\theta_{p,a}) \end{bmatrix}$$

The following equations are used to find the angles used above.

$$\theta = \left| \tan^{-1}\left(\frac{A_y - B_y}{A_x - B_x}\right) \right|$$

$$\theta_{p,b} = \tan^{-1}\left(\frac{p_{b,y}}{p_{b,x}}\right) - \theta$$

$$\theta_{p,a} = \tan^{-1}\left(\frac{p_{a,y}}{p_{a,x}}\right) - \theta$$

A note, since the domain of $\tan^{-1}$ is $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, when the denominator of any of the three previous equations equals zero, the denominator is set to the arbitrarily low value of 0.0001 to approximate the limit of $\tan^{-1}\theta$.

1.1 Implementation

```
void world::resolveCollision(physObject& objA, physObject& objB, float deltaTime) {
    glm::vec3& forceA = objA.forceVectors.add( objB.getID(), glm::vec3(0.0f) );
    glm::vec3& forceB = objB.forceVectors.add( objA.getID(), glm::vec3(0.0f) );

    float angle = glm::abs(glm::atan((objA.pos.y-objB.pos.y)/(objA.pos.x-objB.pos.x)))
   ;

    float momentumAngleA;
    float momentumAngleB;

    if(objA.momentum().x == 0) {
        momentumAngleA = glm::atan(objA.momentum().y/0.0001);
    } else {
        momentumAngleA = glm::atan(objA.momentum().y/objA.momentum().x);
    }

    if(objB.momentum().y == 0) {
        momentumAngleB = glm::atan(objB.momentum().y/0.0001);
    } else {
        momentumAngleB = glm::atan(objB.momentum().y/objB.momentum().x);
    }

    // add other's momentum
    forceA = glm::length(objB.momentum()) * glm::vec3(glm::cos(momentumAngleB - angle)
   , glm::sin(momentumAngleB - angle), 0.0f);
    forceB = glm::length(objA.momentum()) * glm::vec3(glm::cos(momentumAngleA - angle)
   , glm::sin(momentumAngleA - angle), 0.0f);

    // remove own momentum
    forceA -= glm::length(objA.momentum()) * glm::vec3(glm::cos(momentumAngleA - angle
   ), glm::sin(momentumAngleA - angle), 0.0f);
```

```
    forceB -= glm::length(objB.momentum()) * glm::vec3(glm::cos(momentumAngleB - angle
   ), glm::sin(momentumAngleB - angle), 0.0f);

    forceA /= deltaTime;
    forceB /= deltaTime;
}
```