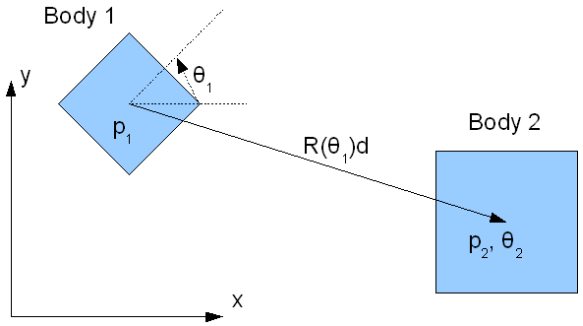
Implementation Note: A Fixed Joint for Box2D - Jorrit Rouwé (July 26st 2009)

In this document we derive the equations needed to implement a fixed joint in Box2D. A fixed joint is a joint whereby 2 bodies are rigidly connected together. This joint can then for example be used to create breakable rigid bodies: At the end of each frame check the constraint force to see if it exceeded a specific maximum value. Please note that a fixed joint is not a very good solution if you just want to connect lots of bodies together as it has a much higher runtime cost and is more unstable than adding multiple shapes to a body.

The code can be found here: www.jrouwe.nl/fixedjoint/Box2D_v2.0.1_fixedjoint_patch.zip

Body 1 is at $p_1(t)$ with angle $\theta_1(t)$, body 2 is at $p_2(t)$ with angle $\theta_2(t)$. For simplicity the center of mass is assumed to be at the position of the object.



The constraint actually consists of 3 constraints packed in one:

- A constraint to fix the distance between the bodies in the x direction
- A constraint to fix the distance between the bodies in the y direction
- A constraint to fix the orientation difference between the bodies

Position Constraint

The position constraint is:

$$\vec{C}_p = \vec{p}_2 - \vec{p}_1 - R(\theta_1)\vec{d}$$

with R being a counter clockwise rotation matrix:

$$R(\theta_1) = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{bmatrix}$$

and the initial vector between the bodies (in the space of body 1) is defined as:

$$\vec{d} \equiv R(\theta_1(0))^{-1}(\vec{p}_2(0) - \vec{p}_1(0))$$

The constraint is satisfied if $C_p = 0$.

Differentiation yields:

$$\dot{\vec{C}}_p = \dot{\vec{p}}_2 - \dot{\vec{p}}_1 - \dot{R}(\theta_1)\vec{d} = \dot{\vec{p}}_2 - \dot{\vec{p}}_1 - A\dot{\theta}_1$$

$$A \equiv \begin{bmatrix} -\sin\theta_1 & -\cos\theta_1 \\ \cos\theta_1 & -\sin\theta_1 \end{bmatrix} \vec{d}$$

We can write this as two equations:

$$\vec{p}_i = (x_i, y_i)$$

$$\dot{C}_x = \dot{x}_2 - \dot{x}_1 - A_x\dot{\theta}_1$$

$$\dot{C}_y = \dot{y}_2 - \dot{y}_1 - A_y\dot{\theta}_1$$

A_x is the x component of A, A_y the y component of A.

Angle Constraint

The angle constraint is:

$$C_a = \theta_2 - \theta_1 - \theta_0$$

with the initial angle difference defined as:

$$\theta_0 \equiv \theta_2(0) - \theta_1(0)$$

again the constraint is satisfied if $C_a = 0$.

Differentiating yields:

$$\dot{C}_a = \dot{\theta}_2 - \dot{\theta}_1$$

Calculating the delta velocity

To calculate the delta velocity for the constraint we need to combine the three equations above into a single equation:

$$\dot{\vec{C}} = \begin{bmatrix} \dot{x}_2 - \dot{x}_1 - A_x\dot{\theta}_1 \\ \dot{y}_2 - \dot{y}_1 - A_y\dot{\theta}_1 \\ \dot{\theta}_2 - \dot{\theta}_1 \end{bmatrix} = 0$$

Using the same notations as the slides for Erin Catto's presentation at GDC09 (http://www.gphysics.com/files/GDC2009_ErinCatto.zip) we derive the Jacobian:

$$\dot{\vec{C}} = J\vec{v} + \vec{b}$$

$$J = \begin{bmatrix} -1 & 0 & -A_x & 1 & 0 & 0 \\ 0 & -1 & -A_y & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}$$

$$\vec{v} = \begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{\theta}_1 \\ \dot{x}_2 \\ \dot{y}_2 \\ \dot{\theta}_2 \end{bmatrix}$$

$$\vec{b} = 0$$

the mass matrix is:

$$M = \begin{bmatrix} m_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & I_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_2 \end{bmatrix}$$

the effective mass as seen by the constraint (a symmetric 3x3 matrix):

$$m_C = (J M^{-1} J^T)^{-1}$$

the Lagrange multiplier (a 3 vector):

$$\vec{\lambda} = -m_C \dot{\vec{C}}$$

which is used to update the velocity to satisfy the constraint:

$$\vec{v}_2 = \vec{\vec{v}}_2 + M^{-1} P = \vec{\vec{v}}_2 + M^{-1} J^T \vec{\lambda}$$

As long as the masses, inertias and θ_1 stay constant, the effective mass matrix will stay the same. It is constant while iterating over the velocity constraints (since only the velocities are updated at that point).

Warm starting

If you accumulate λ over the entire time step as λ_a, you can use it to calculate the initial velocity change

for the next time step:

$$\vec{v}_2 = \vec{\bar{v}}_2 + M^{-1} J^T \vec{\lambda}_a$$

Position Correction

Defining:

$$\vec{x} = \begin{bmatrix} x_1 \\ y_1 \\ \theta_1 \\ x_2 \\ y_2 \\ \theta_2 \end{bmatrix}$$

and combining C into a single equation:

$$\vec{C} = \begin{bmatrix} x_2 - x_1 - (R(\theta_1)\vec{d})_x \\ y_2 - y_1 - (R(\theta_1)\vec{d})_y \\ \theta_2 - \theta_1 - \theta_0 \end{bmatrix} = 0$$

We can use the following equation to correct the positions:

$$\vec{x} = \vec{\bar{x}} - M^{-1} J^T m_C \vec{C}$$

m_C changes only a little bit during the position correction step so we cache θ_1 and check if it has changed enough from the last iteration to warrant recomputation.

Constraining Box2D position instead of center of mass position

So far we have discussed constraining the center of mass between two objects. Box2D makes a distinction between the position and the center of mass. We define the Box2D body positions as P_1 and P_2 which means:

$$\vec{p}_i = \vec{P}_i + R(\theta_i)\vec{CM}_i$$

Where CM_i is the local space vector from the Box2D body position to the center of mass of body i.

And we define:

$$\vec{d}_p \equiv R(\theta_1(0))^{-1}(\vec{P}_2(0) - \vec{P}_1(0))$$

We can then calculate d every time step by:

$$\vec{d} = \vec{d}_p - CM_1 + R(\theta_0)CM_2$$

This means that we can safely alter body shapes without affecting the constraint.