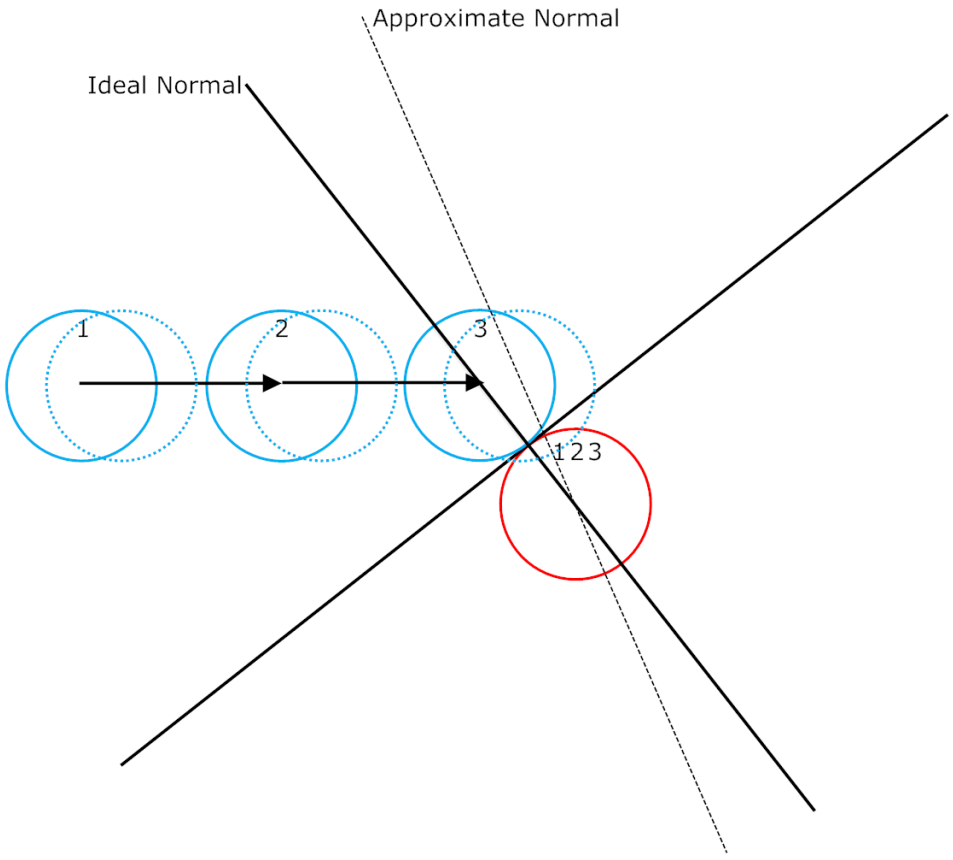
Assignment: A15c (back to main page)

Air-Table: Perfect Kiss

The term *kiss* is taken from the game of billiards (sorry, not very romantic). In the collision calculations that follow this term will refer to the state where two pucks are in contact but not overlapping (no penetration).

As discussed briefly in the *2D-Physics Engine Framework* assignment, there are limitations to establishing the contact normal if based simply on the positions of the overlapping pucks. This overlapped state is when the collision is detected, and it is most straightforward to determine the normal by taking the difference between the position vectors of the two overlapping pucks. This approximate-contact normal is shown as the dotted line in the image below (blue puck approaching a stationary red puck). The best representation of the contact normal is found by reversing the puck positions to the kiss point. The resulting ideal-contact normal is shown with a solid line.



The method for calculating the ideal-contact normal starts with determining the travel-time between the collision detection event and the kiss point. With this travel-time known, the pucks can be reversed to the kiss point using their incoming velocities.

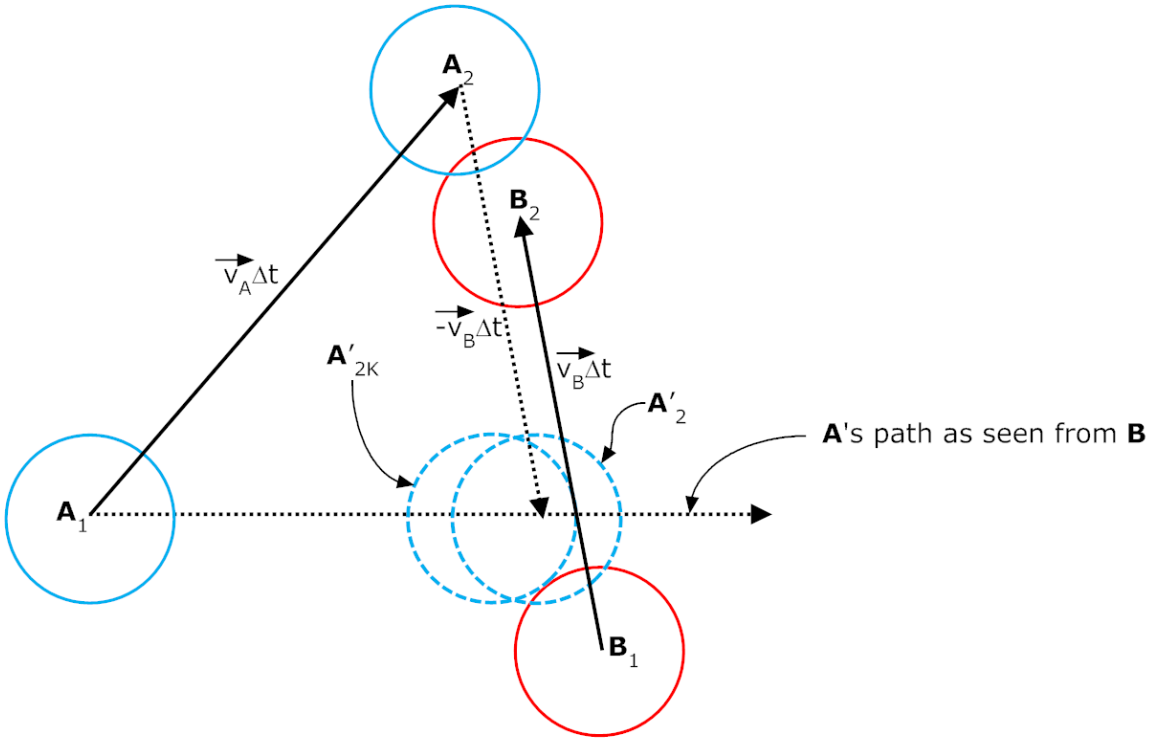
Problem statement:

Add content to the previous exercise to support the perfect-kiss algorithm. Use the 1, 2, and 3 keys on the number pad to launch demos that illustrate a collision with a stationary puck: (1) raw collisions (no overlap correction), (2) overlap correction using the approximate-contact normal, and (3) overlap correction using the idea-contact normal. Simulate with a fixed time-step corresponding to a frame rate of 20 (this fixed time-step will insure evenly spaced pucks result from the persisting drawing). Let all the puck drawing persist (inhibit screen erasing). Use Python's *time.sleep* method to pace the game loop to correspond with the fixed time-step. Use Python's *random.random* method to add variability to the starting position of the moving puck.

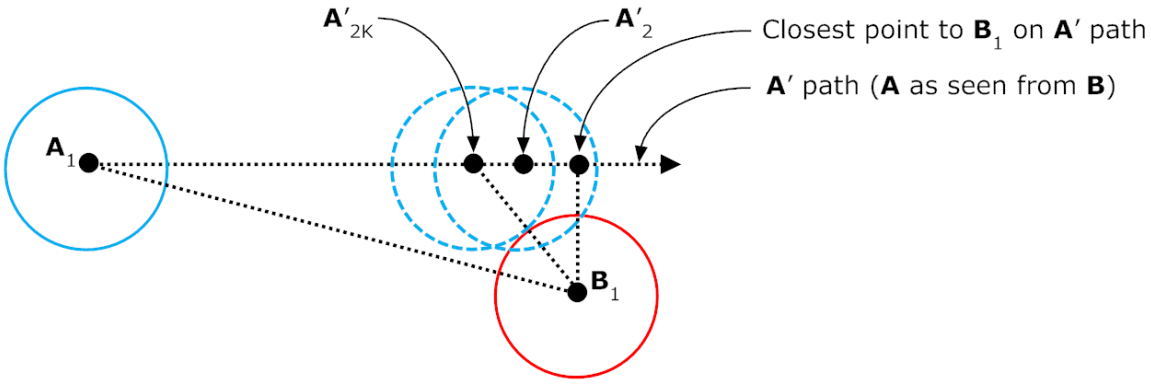
Algorithmic description:

The following outline shows the steps to correct for overlap using the ideal-contact normal.

1. The first drawing shows the two pucks approaching from the A_1 and B_1 positions. The detected collision is shown at the A_2 and B_2 positions. This progression from approach to collision can be observed from the perspective (reference frame) of the B puck. This is done by adding $-\vec{v}_B \Delta t$ to the translation vector for both pucks. This effectively makes B stationary and puts the second position of the A puck at A'_2. From this perspective, the A puck moves along the A' path. This simplifies the problem in that only the motion of A needs to be analyzed to determine the kiss point (when A is at A'_{2K}).



2. Next, determine the legs of the large triangle and the small triangle (inside it) in the image below. This will give the distance that A travels along the A' path between the kiss point and the collision detection point. From this distance the travel time can be calculated using the A' translation vector ($\vec{v}_A \Delta t - \vec{v}_B \Delta t$).
    a. Do this by first projecting the $\vec{B}_1 - \vec{A}_1$ vector onto the A' path. Use the projection method in the vector class.
    b. Subtract the projection vector from the original $\vec{B}_1 - \vec{A}_1$ vector. This difference represents the shortest distance between B_1 and the A' path.



    c. The scalar length of the hypotenuse of the small triangle, by kiss-point definition, must be the sum of the two pucks' radii.
    d. The scalar length of the top leg of the small triangle can be determined using the Pythagorean theorem.
    e. A vector representation of the top leg can be constructed by a product of the unit vector for the A' path and the scalar length of the top segment.
    f. Now, the vector from A_1 to A'_{2K} can be determined (vector to the kiss position). From here it is straightforward to calculate the vector between A'_{2K} and A'_2.
3. From this A'_{2K} to A'_2 travel, the corresponding travel time can be calculated using the A' velocity vector ($\vec{v}_A - \vec{v}_B$). This can be done with either the *x* or the *y* component; just avoid zero in the denominator. This gives a key result: t_p, the time that transpires between the kiss point and the collision detection (the subscript *p* refers to penetration).
4. Once t_p is determined, we can go back to the absolute reference frame and use the original pre-collision velocities of the two pucks ($\vec{v}_A$, $\vec{v}_B$). Use these absolute velocities and t_p to reverse the A_2 and B_2 puck positions back to the kiss point.
5. Now, at the kiss point, determine the ideal-contact normal and the post-collision normal velocities. Add the tangential components to get the total post-collision velocity vectors.

6. Use the post-collision velocities to move the pucks forward to a time t_p after the kiss point. This puts the pucks where they should be had they collided at the surface and not penetrated each other.