STUDY OF COLLISION DETECTION TECHNIQUES FOR 2D OBJECTS

by

Mugdha Barve
Pradeep Jayaraman

ABSTRACT
We use this paper as a means to discuss different approaches for detecting frictionless, elastic collisions between objects. We explain the derivations to predict the time to first collision, checking static interferences and computing new velocities of discs after collision in 2D animation. We compare two approaches : PIT (Periodic Interference Test) and PIC (Predicted Instance of Collision), and present specific conditions under which PIC proves as a better option over PIT. We also discuss the cases under which PIT may be considered over PIC.

1.1 STATIC INTERFERENCES:

In computer graphics the term intersection test, usually refers to the task of determining if two geometric primitives intersect at a specified discrete instant in time. Consider the case of determining whether two implicitly defined spheres intersect. This can be done by checking the distance between the centers against sum of the radii.

1.2 ELASTIC COLLISIONS:

An elastic collision in an isolated system is a collision in which kinetic energy is conserved. Elastic collisions occur when forces between the colliding bodies are conservative. For example, when two steel balls collide, they squash a little near the surface of contact, but then they spring back.

Consider two particles, denoted by subscripts 1 and 2. Let m be the mass, u be the velocity before collision and v be the velocity after collision. The total kinetic energy is the same before and after the collision. Hence,

½ m1u1^2 + ½ m2 u2^2 = ½ m1v1^2 + ½ m2 v2^2 --> equ. (1)

The total momentum remains constant throughout the collision. Hence,

m1u1 + m2 u2 = m1v1 + m2 v2 --> equ. (2)

We have two equations, and two unknowns, namely v1 and v2. Solving them simultaneously, we get the solution,

v1 = u1 ((m1 - m2) + 2 m2 u2)/ (m1 + m2)
v2 = u2 ((m2 – m1) + 2 m1 u1)/ (m1 + m2)

For sake of simplicity,
Equation 1 can be rewritten as,

$$m1u1^2 - m1v1^2 = m2\,v2^2 - m2\,u2^2$$
$$m1(u1 - v1)(u1 + v1) = m2\,(v2 - u2)(v2 + u2) \quad \text{--> equ. (3)}$$

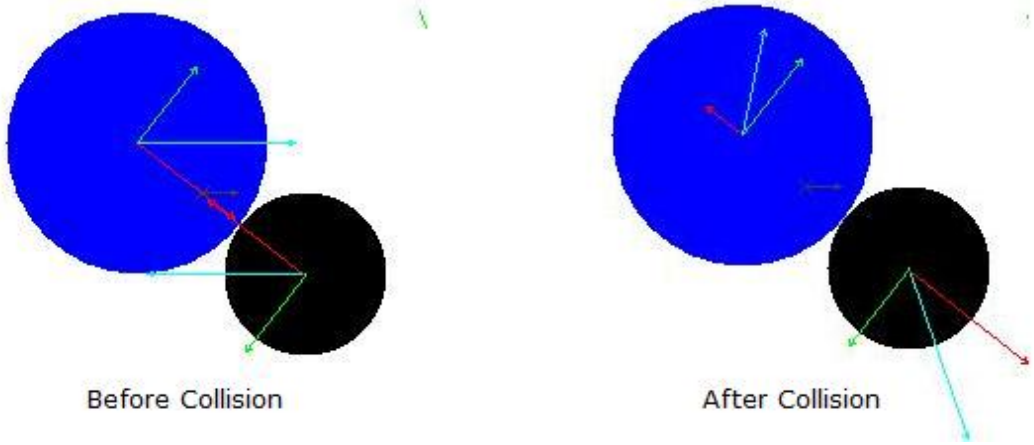
Equation 2 can be rewritten as,

$$m1(u1 - v1) = m2(v2 - u2) \quad \text{--> equ (4)}$$

Diving equ (3) by equ (4), we get

$$u1 + v1 = v2 + u2$$

So once one of the final velocities are computed, the other can be easily obtained by using the above equation.



Before Collision :
The blue arrow denotes the direction in which the balls travel before the collision. And its length denotes the magnitude of the velocity.
At the instant of collision, the velocity is split into tangential and normal components. One of the components is taken along the line joining the centers. And the other is perpendicular to this component.

After Collision :
The green vectors (normal to the line joining the centers of circles) would not have changed after collision, as long as friction is not considered. The green components do not affect each other as they act in opposite directions. The resulting vectors along the line joining the centers are computed and their magnitudes are determined. With the help of the normal and tangential vectors, the new resultant velocity (blue vectors) can be computed using Pythagoras theorem.

1.3 COLLISION DETECTION:

The static interference test is carried out by comparing distance between the two centers. If it results to be less the to sum of the radii, then it means collision has occurred.

$$||c2 - c1|| - (r1 + r2) = 0$$

If PIC technique is adopted, then the time factor needs to be considered. So the equation is rewritten as

((c2 + tv2) - (c1 + tv1))^2 = (r1 + r2)^2

where v1 and v2 are the velocity at which the circle moves. The equation is squared on both sides so as to remove the square root and make computation easier.

1.3.1 PIT – Periodic Interference Test:

It is the most simple and straightforward way for detecting collision between two objects. For each frame the object is moved, collision test is carried out to find if interference happens at the current frame. If it returns positive, then its motion is stopped right there, or sometimes even reverted back to its previous frame's position. Though this might be easy to implement, it is highly flawed because it detects collision only after the event has occurred.

Another problem is that, sometimes an objects previous state or position may no longer be true. So it would not be possible to return it to its previous frame before collision occurred.

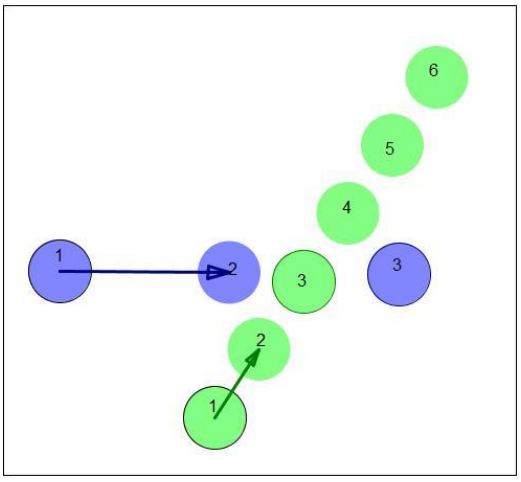
1.3.2 PIC - Predicted Instance of Time

This approach overcomes the drawbacks of PIT. The reason that PIT fails to be accurate is that, it is able to check for collision only at the advent of each frame, and has no information about what happens in between frames. On the other hand, PIC computes the exact time of collision between the current frame and the next upcoming frame. And unlike PIT, instead of advancing to the next frame, it advances to the intermediate frame when collision has just occurred, thereby avoiding the occurrence of overlap.

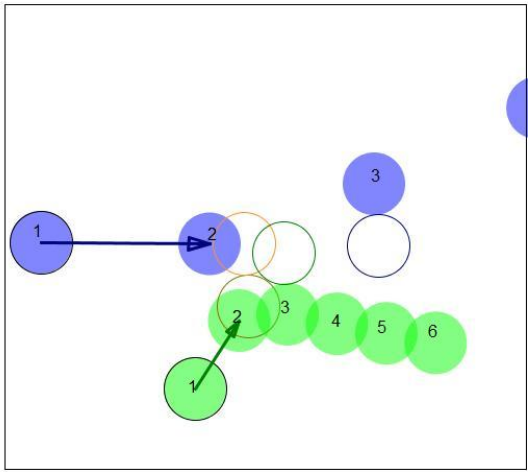
2. ERRORS INTRODUCED DUE TO PIT:

The following examples show how PIT can introduce erroneous behavior while detecting collisions:

CASE 1 : COLLISION MISSED in PIT



USING PIT

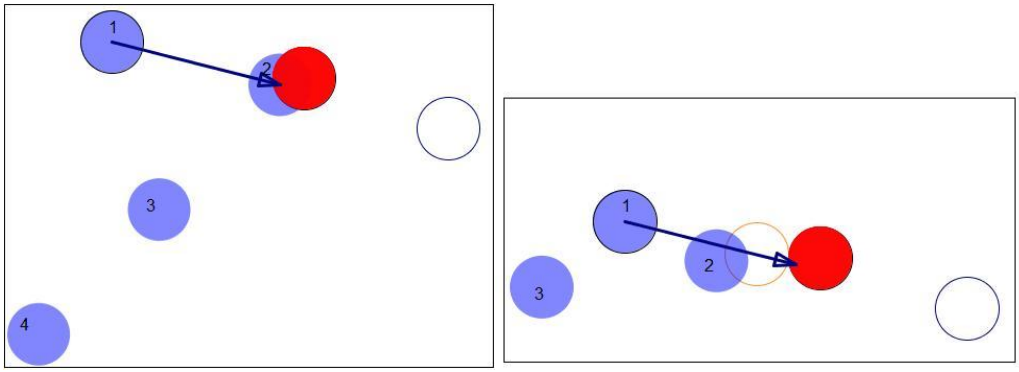


USING PIC

This case shows how a collision may be completely missed when using PIT. The first figure uses PIT to compute the occurrence of collision. As the frame rate is low, the collision which would have occurred between frame 2 and 3 of the blue ball is completely missed and both balls move

without affecting each other.
As shown in the second figure, since PIC is used, it checks for a collision between the 2 balls for every next frame. In this case, it detects one between frame 2 and 3 of the blue ball, calculates the time of collision, proceeds the animation to that point, calculates resulting velocities and continues the animation with the new velocities.

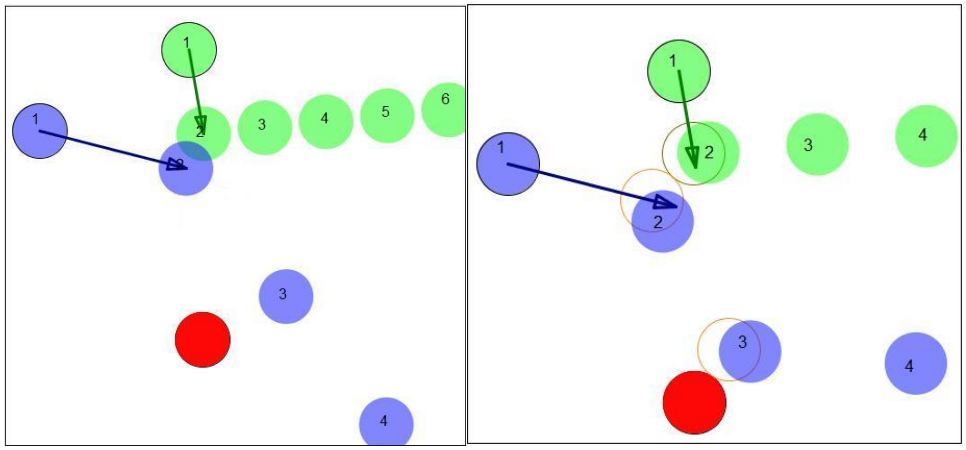
CASE 2: WRONG DIRECTION AFTER SHOCK in PIT



USING PIT USING PIC

This case illustrates how the use of PIT can result into calculation of the wrong directions after collision. As shown in figure 1, due to PIT, the collision is detected when the balls are already intersecting each other. The directions of the balls after collision are calculated with error since the vector joining the centers of the discs changes.
The second figure uses PIC and detects the collision in between two frames. The unfilled outline of the blue ball indicates the position between frame 1 and 2 where the collision actually takes place. So it calculates the directions and velocities of the balls from that point. The second figure shows the actual direction of the balls after the collision.

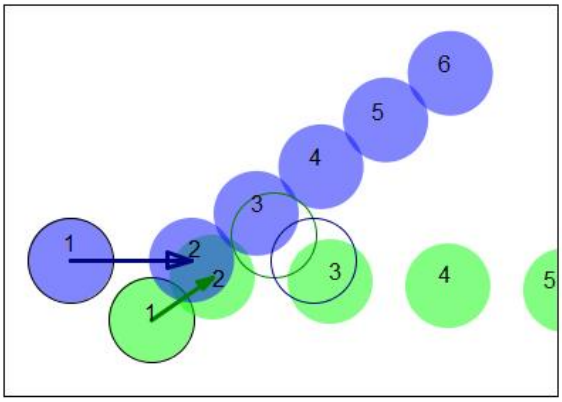
CASE 3: EFFECTS ON MULTIPLE COLLISIONS



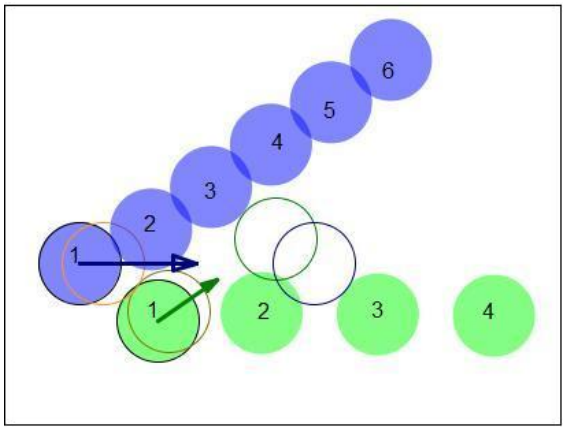
USING PIT USING PIC

This example illustrates how PIT can affect multiple collisions. As shown is first figure, due to use of PIT, the collision of the balls at frame 2 is detected a bit late. Thus, the blue ball entirely misses the next collision with the static red ball. The second figure shows how the multiple collisions should have occurred. The unfilled outlines of the balls indicate where the balls have actually collided.

CASE 4: DELAY



USING PIT



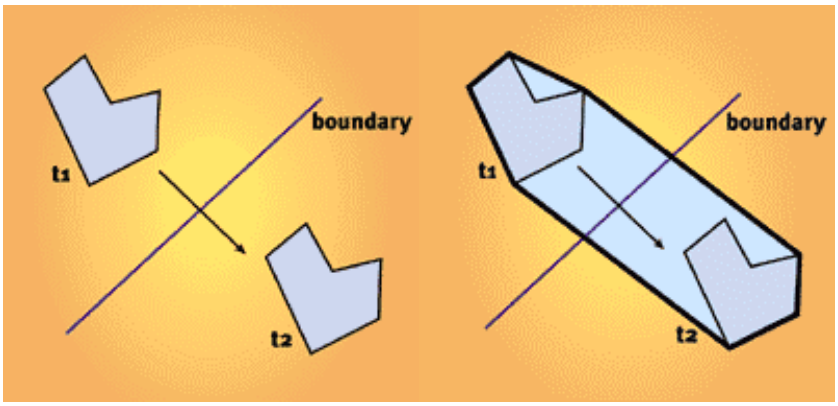
USING PIC

The above figures illustrate how PIT can cause delayed actions after collision. Comparing the first and second figure it can be observed that after collision is detected by their respective methods, the balls in both the figures seem to go in almost the same direction. But there is a time delay introduced in the first figure i.e. when using PIT to detect collision. This is because PIT detects the collision after the balls intersect each other. The second figure shows where the collision would have been detected (indicated in unfilled blue and green intersecting circles) if PIT was used instead of PIC.

3.1 ALTERNATE PIC:

Solving equations to determine the exact time of collision might be easy for simple objects, but is not so for complex objects. In such cases, the time between 2 frames can be recursively split to check in which half the collision occurs. However, subdividing the given time interval into half and testing for intersection at the midpoint each time could prove costly. Instead we could create a solid(or simply a convex hull or bounding box) out of the space that the original object occupies between time t1 and t2 and then test the resulting solid against other object.

Instead of restricting the prediction of collision time to next frame alone, the actual time of collision at any point future can be computed. This waives the need to check for collision at each frame. However, it will hold good only if the object retains it state throughout till the collision occurs.



3.2 ALTERNATE COLLISION DETECTION TECHNIQUES:

If we have a lot of objects in the scene, pair wise interference testing might become costly. The number of comparisons required is O(N2). But this can be avoided in several ways. For instance, we can divide our world into objects that are stationary (collidees) and objects that move (colliders) even with a v=0. For example, a rigid wall in a room is a collidee and a tennis ball thrown at the wall is a collider. We can build two spatial trees (one for each group) out of these objects, and then check which objects really have a chance of colliding. We can even restrict our environment further so that some colliders won't collide with each other — we don't have to compute collisions between two bullets, for example.

Another method is to use Portals. Portal-based engines divide a scene or world into smaller convex polyhedral sections. Convex polyhedra are well-suited for the graphics pipeline because they eliminate overdraw.

A table containing the distance between each object with every other object in the scene can be maintained. The table needs to be updated only when any object moves. Static objects like a wall, which remain constant, can be easily managed. But the downside is, when a single object moves, its distance with all other object needs to be computed again.

The area of the object under observation can be pre-computed. At each frame of moment, the area is recalculated by determining its interior boundaries. When a collision occurs, the area would result to be less than the pre-calculated value.

4. CONCLUSION:

Favourable conditions for PIT:
The cases considered above discuss the different types of errors introduced in an animation when PIT is used for collision detection. PIT causes problems such as missing collisions when the frame rate is very low or if the colliding objects have small dimensions. Problems such as delays, computation of wrong directions are caused due to low frame rates.
Thus, PIT may be acceptable in situations where the frame rate is high and the objects whose collisions are being detected are not very small in dimension. If the frame rate is high, and assuming that the objects moving in the animation are not moving very fast (relative to the frame rate), then each object will move by a fairly small distance in each subsequent frame. So, even if PIT is used, the amount of intersection occurring between objects will be very low. Thus errors introduced will be negligible.

Unfavourable conditions for PIC:
Clearly, Predicted Instant of Collision (PIC) is more accurate compared to the Periodic Interference Test(PIT). However, the cost involved in making the computation has to be scrutinized. In cases where subjects of collision are simple shapes, the complexity involved in checking for collision is less, but once complex shapes like a concave polygon or a group of polygons have to be checked, the computation cost becomes heavy and undesirable. The whole process becomes more intricate if the objects are subjected to rotation and other transformations.
This problem can be alleviated somewhat by introducing convex hulls or bounding boxes around the polygon to make calculations simple. But then, accuracy is compromised, which is exactly what PIC is trying to concentrate on.
This is when the aim of checking for collision needs to be studied. If the place is video games, then emphasis is more on real time responses rather than on the accuracy of collision. But it is not the case for simulations or animated movies.

Thus the technique used to detect the collision between objects can be chosen depending on the type of application and the geometry of the objects under consideration.

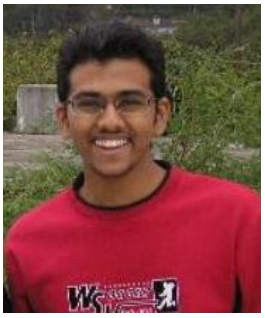
REFERENCES:

- Collision Detection and Proximity Queries SIGGRAPH 2004 Course
  Authors : Sunil Hadap R&D Staff - Dynamics, PDI/Dream Works
  Dave Eberle R&D Staff - Dynamics, PDI/Dream Works
  Pascal Volino University of Geneva
  et al.
  http://portal.acm.org/citation.cfm?id=1103900.1103915

- Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies by David Baraff

- http://www.phy.ntnu.edu.tw/ntnujava/index.php?topic=4

AUTHORS:



PRADEEP JAYARAMAN
http://www.prism.gatech.edu/~pjayaraman3/cs6491

MUGDHA BARVE
http://www.prism.gatech.edu/~mbarve3/CS6491/