Rock System

1. Objective

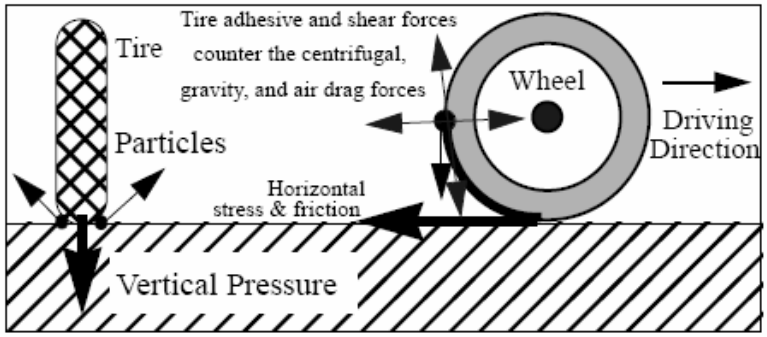
To develop a bed of rocks to follow the player vehicle so that other vehicles in front of the player can shoot those rocks into the air. The vehicles will launch the rocks backwards. For our purpose, collision only occurs between the rocks and the rear vehicle tires.

The bed of rocks is a circle that is centered on the player vehicle. Rocks, which moved out of the radius of the circle, will get repositioned at the leading edge of the circle.

2. Design Considerations

Rocks Behavior

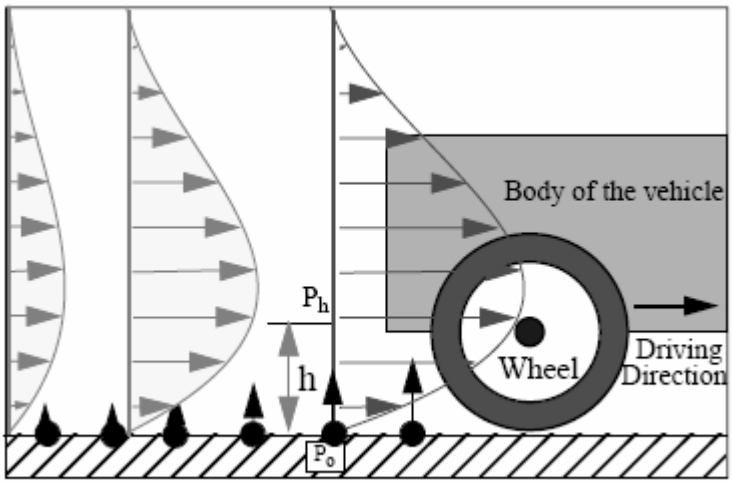
Like sand particles, rocks are small and granular materials. I collectively consider them as dust particles. Based on a paper, there are two reasons why a fast moving vehicle splashes particles into the air.



a) Particles generated around the tire

Taken from [1]

When the tires roll over the particles, they stick to the tire surface due to tire adhesive forces and frictions. The stuck particles ejects off the tire at different height of the tire surface based on parameters such as tire velocities, tire angular velocities, ground wetness and particle weight. In particular, tire velocities are factored in. It helps to simulate the behavior of rocks splashing from side of the tires when the vehicle is doing a sharp turn.



b) Particles generated underneath and behind the vehicle

Taken from [1]

The pressure gradient, formed by the moving air underneath and behind the vehicle, will lift up fine particles. This can be ignored since this phenomenon has insignificant effect on rocks.

Efficient Collision Detection

A large number of rocks are expected and they are located anywhere in an unordered manner.

In order to reposition the rocks that are out of the rock bed radius, every rock position requires constant monitor. Conventional collision detection techniques arrange collision objects in some spatial order so that we can efficiently identify the intersecting objects with respect to another object.

One example is the bounding volume hierarchy (BVHs). BVHs are often represented by tree structures. However, it is not straightforward to find objects that are far away from a reference point or object. Furthermore, it may be expensive to update the tree hierarchy when the rock objects are repositioned.

The collision and sweep method loops through a sorted list of collision objects and perform AABB intersection test. Unfortunately, the technique relies on temporal coherence. The rocks are frequently repositioned at random location and it becomes more computationally expensive to resort the list.

The third form is spatial subdivision. If an object is found at some location in the world, it will be mapped to one position in a spatial data structure. 3 common data structures are binary space partition trees (BSPs), quad- and oct-trees, and grids. It is worth noting that the grid data structure is static to the world. If an object has relocated or re-orientated, the grid doesn't change its structure. Recall

that you have to update the tree hierarchy if the rock positions change. Mapping an object to a grid cell incur little cost. Thus, it is suitable in situations with large and frequent spatial re-ordering.

3. Method

This section will develop the implementation of the rock system. The main feature of the rock system is the use of spatial hashing to support efficient collision detection for the rocks and vehicle tires.

The rocks are hashed into a hash table based on the minimum and maximum values of their AABB. This implicitly subdivide the world space into 3D cells and the hash function maps the 3D cells into a 1D hash table index. This is similar to the function of the grid data structure that was discussed in the previous section.

As a result, each hash table index only contains a small number of rocks that have to be involved in intersection tests with the tires.

The operations of the rock system can be outlined in 2 passes:

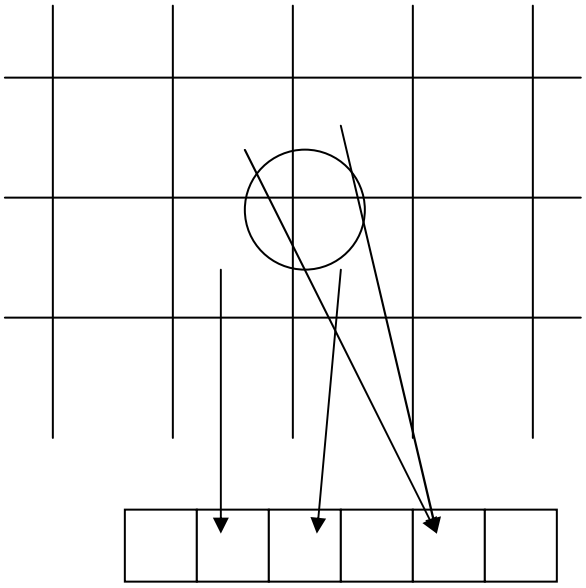
Pass 1

- For each rock,
- Check distance of the rock from the rock bed center.
- If the distance is greater than bed radius, reposition the rock at leading edge of rock bed.
- Hash the rock into the respective hash table index base on the 3D cells it occupies in world space.

Pass 2

- For each vehicle rear tire,
- Identify the 3D cells it occupies and the corresponding hash indices.
- Perform collision detection between the tire and rocks found in the identified hash indices.
- If collision has occurred, give the collided rock an initial impulse.

Spatial Hashing



In the first pass of the rock system, the AABB of the rocks are discretized into 3D cells and the rock is hashed into the respective hash index of the 3D cells. A world position (x, y, z) is mapped to a 3D cell with index (i, j, k). The mapping function is given as:

$$i = \lfloor x / l \rfloor, j = \lfloor y / l \rfloor, k = \lfloor y / l \rfloor$$

Where:

- *l* is the cell grid size and chosen to be the smallest possible radius of rock

The hash function is given as:

Hash(i,j,k) = (i p1 **xor** j p2 **xor** k p3) mod n

Where:

- The value n is the hash table size.
- p1, p2 and p3 are some large prime numbers

Intersection Test

In pass 2, each tire is represented by a tire ray that runs through the tire bottom and tire center. The hash indices are found for all grid cells intersected by the ray. All rocks found at the according hash indices are tested using the ray-sphere intersection test.

Collision Response

Once a rock is intersecting a tire ray, the only thing left to do is to give it an initial impulse.

The vertical impulse (Z direction) of the rocks is affected by the static friction of the rock and the angular velocity of the tire.

The impulse in the XY direction is affected by the tire velocities.

Reference

1. Chen, J., Fu, X. and Wegman, E.J. Real-Time Simulation of Dust Behavior Generated by a Fast Traveling Vehicle. *ACM Transactions on Modeling and Computer Simulation*, *9* (2). 81-104.