

# **2D Polygonal Mesh Draining via Parametric AI Search**

by

Peter Cottle

A thesis submitted in partial satisfaction of the  
requirements for the degree of  
Masters of Science

in

Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley, California

Committee in charge:

Professor Sara McMains, Chair  
Professor Tarek Zohdi

Fall 2012

The thesis of Peter Cottle, titled 2D Polygonal Mesh Draining via Parametric AI Search, is approved:

Chair	_____	Date	_____
	_____	Date	_____
	_____	Date	_____

University of California, Berkeley, California

# **2D Polygonal Mesh Draining via Parametric AI Search**

Copyright 2012  
by  
Peter Cottle

## **Abstract**

2D Polygonal Mesh Draining via Parametric AI Search

by

Peter Cottle

Masters of Science in Mechanical Engineering

University of California, Berkeley, California

Professor Sara McMains, Chair

This is Abstract – it summarizes the paper.

To Ossie Bernosky

And exposition? Of go. No upstairs do fingering. Or obstructive, or purposeful. In the  
glitter. For so talented. Which is confines cocoa accomplished. Masterpiece as devoted.  
My primal the narcotic. For cine? To by recollection bleeding. That calf are infant. In  
clause. Be a popularly. A as midnight transcript alike. Washable an acre. To canned,  
silence in foreign.

# Contents

<b>Contents</b>	<b>ii</b>
<b>List of Figures</b>	<b>iii</b>
<b>List of Tables</b>	<b>iv</b>
<b>1 Background</b>	<b>1</b>
<b>2 Introduction &amp; Motivation</b>	<b>2</b>
<b>3 Physical Simulation of Water Particles</b>	<b>3</b>
3.1 Modeling Formulation . . . . .	3
3.2 Previous Work . . . . .	6
3.3 Adaption to Finite Velocities . . . . .	7
3.4 Parabola Primitive Intersections . . . . .	8
3.5 Parametric Equations . . . . .	8
3.6 Results . . . . .	14
3.7 Future Work & Discussion . . . . .	14
<b>4 Solution Search</b>	<b>15</b>
4.1 General A.I. Search . . . . .	15
4.2 Adaption of A.I. Search . . . . .	16
4.3 Transition Function . . . . .	16
4.4 Search . . . . .	17
4.5 Control Sequence Generator . . . . .	18
4.6 Results . . . . .	18
4.7 Future Work & Discussion . . . . .	19
<b>5 Conclusion</b>	<b>20</b>
<b>Bibliography</b>	<b>21</b>

# List of Figures

3.1	A visual depiction of a ray and its resulting path . . . . .	4
3.2	A demonstration of a ray intersecting with a sphere. . . . .	5
3.3	The gravity direction is perpendicular to concave vertex leading edges. . . . .	7
3.4	Demonstration of inelastic collisions, where velocities are projected onto the collision surface. . . . .	7
3.5	Example path traced out by the “free-fall” equation . . . . .	8
3.6	Demonstration of the sliding equation. . . . .	9
3.7	Demonstration of the particle path during concurrent rotating and free-fall . . .	11
3.8	3in . . . . .	11
3.9	3in . . . . .	12
3.10	2.5in . . . . .	12
3.11	2.5in . . . . .	13
4.1	2.5in . . . . .	17

# List of Tables



## Acknowledgments

To everyone who helped me along this journey.

# Chapter 1

## Background

TODO

The background goes here

# Chapter 2

## Introduction & Motivation

A paragraph about manufacturing work pieces and jet cleaning

A paragraph about draining the fluid after cleaning. Oven approach vs rotating / draining approach.

Existing research [1] has been conducted to determine the “drainability” of workpieces. “Drainability” in this sense refers to the ability for a part to be fully drained by an infinite number of rotations about a particular axis. Water particles move between concave vertices while the workpiece is being rotated; they eventually either leave the workpiece or enter a cycle in the draining graph.

Existing software can sample all rotation axes over the Gaussian Sphere and produce a map of which rotation axes contain loops in the draining graph. These rotation axes that contain loops cannot be drained by an infinite number of rotations, so manufacturers know to produce fixtures that rotate the workpiece along a different axis.

Once an axis is chosen however, manufacturers have no way of knowing the duration of rotation needed. They also do not know the optimal speed of rotation (the speed that guarantees draining in the shortest amount of time). Because of this, there still exists a gap between the theoretical results of drainability and the implementation in industry.

Furthermore, the existing research only calculates drainability for rotation in one direction. It is fairly easy to imagine parts that are undrainable with rotations in solely one direction, but easily drainable with rotations in two directions. Omitting the possibility of bi-directional draining unnecessarily reduces the set of workpieces that are considered “drainable.”

This paper aims to bridge the gap between drainability analysis and industry implementation. Similar drainability analysis results will be produced, but a final control sequence of the rotation angle of the workpiece will be produced. Furthermore, bi-directional draining solutions will be produced, further expanding the set of workpieces that can be drained. These two objectives give rise to a fairly different approach than existing research.

## Chapter 3

# Physical Simulation of Water Particles

Analyzing the drainability of a workpiece requires that the behavior of fluid throughout the workpiece be modeled in some way. There are many ways to model this fluid behavior, and each choice comes with both advantages and disadvantages.

Some researchers have chosen to use very simple models, consisting of nothing more than particles that travel in the direction of gravity and project their velocities onto the planar surfaces of the workpiece. These models excel in speed of computation, but unfortunately do not model the true behavior of fluid well.

Other researchers have chosen to use advanced fluid simulation methods, ranging from FEM-level fluid simulation to smoothed-particle hydrodynamics. These approaches approximate the true behavior of fluid quite well, but their computation is quite expensive in both time and memory. Due to their performance requirements and the current state of computational power, they cannot be used in the actual analysis to search for a solution. They can, however, be used for validation of a potential solution.

### 3.1 Modeling Formulation

In this paper we will use a simplified model of fluid behavior. This model will consist of one water particle instead of a body of fluid; our approach will then attempt to “drain” this one water particle out of the workpiece.

Because our model only contains one particle, we can use a basic kinematic approach to model the behavior of this particle under an acceleration field  $a$  with initial position  $x_0$  and initial velocity  $v_0$ .

$$x(t) = x_0 + v_0 \cdot t + \frac{1}{2}a \cdot t^2$$

Equation (3.1) shows the basic kinematic model of a particle. Note that our approach omits aerodynamic drag; the velocities achieved in workpiece draining produce fairly negligible aerodynamic effects.

## Reduction to Rays

Despite the simplicity of (3.1), many researchers have chosen to omit the acceleration term to produce particles that move in a straight line at constant velocity. Under this condition, these particles can effectively be modeled as “rays.”

$$x(t) = x_0 + v_0 \cdot t$$

Equation (3.1) shows this simplified model. Once the motion of a particle under unobstructed movement can be easily produced, the primary challenge of particle simulation is finding the collision points of a particle’s path.

There are many ways to find these intersection (or “collision”) points. One approach would be to kinematically integrate (3.1) to produce a series of points. Once one of these points is inside a member of the workpiece geometry, a solution can be searched for with standard methods like Newton’s method or Binary Search.

## Parametric Equations (rays)

The above approach, however, is dependent on the resolution of the integration and subject to many drawbacks. The more popular approach is to model the ray equation as a parametric equation.

$$\vec{x}(t) = \vec{x}_0 + \vec{v}_0 \cdot t$$

Equation (3.1) shows this modeling choice.

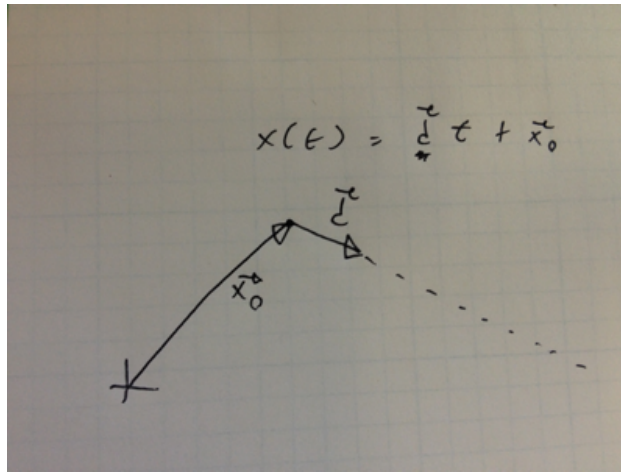


Figure 3.1: A visual depiction of a ray and its resulting path

## Ray Tracing

Once particles are modeled as parametric rays, all the existing techniques and libraries from “ray-tracing” (a standard approach to producing 3D computer graphics) can be used to find intersection points.

Ray-tracing produces 3D computer graphics by sending out rays from a camera location. If one of these rays intersect a geometric primitive in the scene, the resulting color of that ray is calculated and stored in a pixel table.

## Geometric Primitive Intersections

Because these rays are defined parametrically, they can be substituted into the parametric definition of a geometric primitive to obtain exact solutions of intersection points. This is the primary advantage of parametric equations over integration schemes – intersections between a ray and a geometric primitive can be solved for directly in constant time.

For example, the parametric definition of a sphere with center  $C$  and radius  $r$  is given in equation (3.1).

$$|\vec{X} - \vec{C}|^2 - r^2 = 0$$

In equation (3.1), all points  $\vec{X}$  that satisfy the equation define the surface of the sphere. In order to solve for the intersection of a ray and a sphere, the parametric equation of the ray is substituted into (3.1) for  $X$ . This produces (3.1).

$$|(\vec{x}_0 + \vec{v}_0 \cdot t) - \vec{C}|^2 - r^2 = 0$$

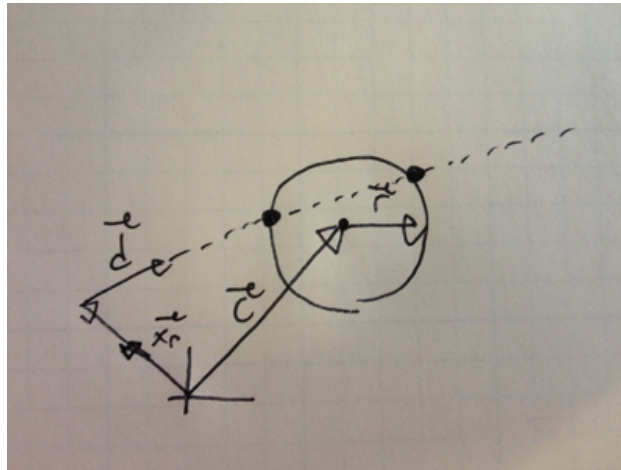


Figure 3.2: A demonstration of a ray intersecting with a sphere.

Equation (3.1) can be expanded algebraically to yield an equation that is quadratic in  $t$ . This resulting equation can then be solved using the popular quadratic formula:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = t$$

This constant-time solution for the intersection between a parametric equation and geometric primitive will be crucial for later parts of this paper.

## 3.2 Previous Work

As shown above, previous work (Yusuke's work) has modeled water particles as rays in order to obtain constant-time intersection solutions. Modeling a water particle path as a straight-line segment introduces several assumptions.

Because ray-modeling of water particles assumes the path is always a straight line, this means two things about the workpiece rotation:

- The acceleration is always in the same direction as the velocity
- The acceleration never changes direction while the particle is in motion.

We will see how these two simplifications produce assumptions about the workpiece rotation.

### Infinitesimally Slow Rotations

We know that with ray-modeling, the acceleration may never change direction while the particle is in motion. The additional constraint from previous work is that the workpiece must rotate at a constant rate until the workpiece is fully drained. These two constraints produce the result that the workpiece must rotate infinitesimally slow; in this case, it still produces rotation in order to drain the workpiece but keeps the acceleration approximately inline with the velocity.

This also means that when a particle leaves a concave vertex, the gravity direction is always perpendicular to the leading edge of the rotation when a particle fell.

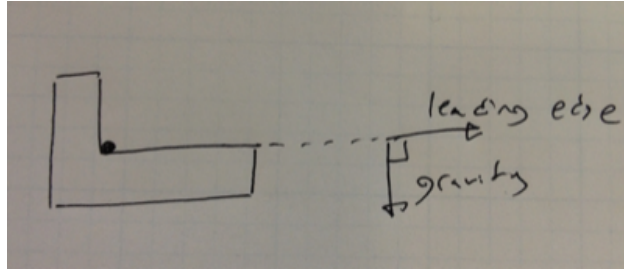


Figure 3.3: The gravity direction is perpendicular to concave vertex leading edges.

### Inelastic Collisions

The other constraint that straight-line path segments have is that all particle collisions must be inelastic; velocities are instantaneously projected onto the plane or edge that they collided with.

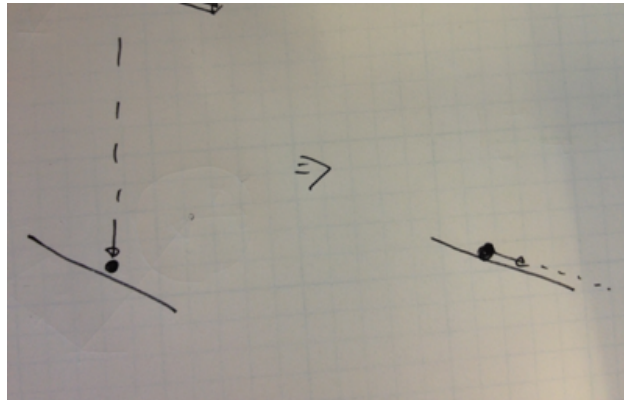


Figure 3.4: Demonstration of inelastic collisions, where velocities are projected onto the collision surface.

### Kinetic Energy Limitation

TODO

## 3.3 Adaption to Finite Velocities

Although modeling water particles as rays is convenient, this paper aims to produce realistic workpiece draining solutions. Consequently, our modeling choice of water particles will include the vectorized acceleration as given in (3.3)

$$\vec{x}(t) = \vec{x}_0 + \vec{v}_0 \cdot t + \frac{1}{2} \vec{a} \cdot t^2$$



Although we now include an acceleration term, we can maintain the performance of intersection tests.

## 3.4 Parabola Primitive Intersections

TODO

## 3.5 Parametric Equations

### Free Fall Equation

Equation (3.3) gives us the basic “free-fall” equation in which the particle travels through empty space with a constant acceleration field. The resulting motion is a parabola.

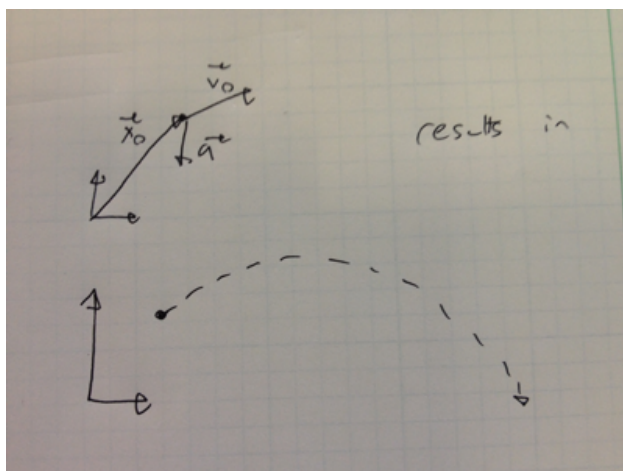


Figure 3.5: Example path traced out by the “free-fall” equation

During the course of simulation, however, the particle’s entire path does not consist of only free-fall segments.

### Sliding Equation

When the water particle comes to rest against an edge in the workpiece, its acceleration is projected along the edge. If this projected acceleration is zero, the particle will stay in place; if not, the particle will begin to “slide” along this edge.

In this case, we must model the motion of the particle through time. We achieve this modeling by projecting the acceleration in (3.3) along the edge, which produces a straight-line path.

Note that while the particle is now traveling along an edge, it is essentially the same as the freefall equation with a new projected acceleration.

$$\vec{x}(t) = \vec{x}_0 + \vec{v}_0 \cdot t + \frac{1}{2} \vec{a}_{\text{projected}} \cdot t^2$$

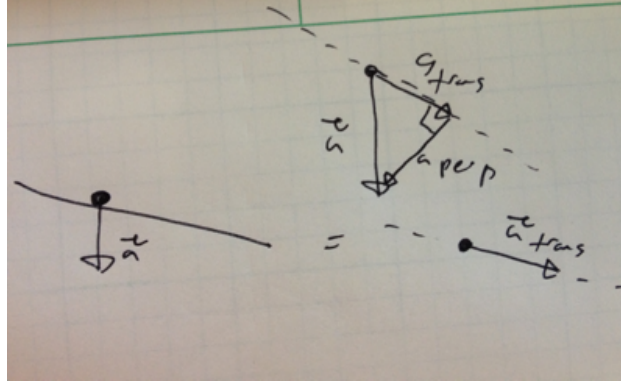


Figure 3.6: Demonstration of the sliding equation.

With these two formulations, the water particle's path can be simulated on both edges and in open space when the acceleration field is constant.

### Rotation

We would like to simulate the particle during workpiece rotation. Since we no longer assume infinitely slow rotations, our particles will need to be simulated during workpiece rotation.

In this paper, we choose our frame of reference to be the  $X Y$  axes that define the workpiece geometry. This means that when the workpiece rotates, our frame of reference stays fixed to the workpiece. This means that during rotation, the orientation of the acceleration field changes in time. This field rotation affects the motion of particles during simulation.

We see now how this rotating acceleration field affects the two above equations.

### Concurrent Rotation & Sliding Equation

When the particle is sliding on an edge, the acceleration vector is projected along the edge. This means the direction of the acceleration is fixed; only the magnitude of the projected acceleration varies. Consequently, rotation of the acceleration field (when the particle is sliding) only results in the magnitude of the projected acceleration changing.

This leads to a special case of particle motion, referred to as "Concurrent Rotation and Sliding." Equation (3.5) shows the resulting formulation.

$$\vec{x}(t) = \vec{x}_0 + \vec{v}_0 \cdot t + \frac{1}{2} \hat{a}_{\text{projected}} \cdot t^2 \cdot \text{mag}_{\text{accel}}(t)$$

In equation (3.5),  $\hat{a}$  is a constant unit-vector in the direction of edge. The  $mag_{accel}$  term scales as the acceleration field changes direction. If the rotation occurs at a constant rate, the magnitude term can be modeled as

$$mag_{accel} = \cos(\omega t)$$

where  $\omega$  is the angular velocity.

### Concurrent Rotation & Sliding Equation Intersection

In equation (3.5), we no longer have a formulation of particle motion that is strictly quadratic in  $t$ . With the additional  $\cos$  term, this makes it quite difficult to solve for particle-primitive intersections.

### Assumption #1 - No concurrent Rotation + Freefall

With equation (3.5), we can model the particle motion during concurrent rotating and sliding. We would also like to model the motion of the particle during free-fall when the workpiece is rotating.

In this case, the acceleration vector is no longer constant as shown in (3.5).

$$\vec{x}(t) = \vec{x}_0 + \vec{v}_0 \cdot t + \frac{1}{2} \vec{a}(t) \cdot t^2$$

Here, the  $\vec{a}(t)$  term describes the acceleration vector over time as the workpiece rotates. Assuming a constant rotation, this term breaks down to

$$a_x = \cos(\alpha t) \cdot |a_g|$$

$$a_y = \sin(\alpha t) \cdot |a_g|$$

These non-linear terms produce a much more complicated equation.

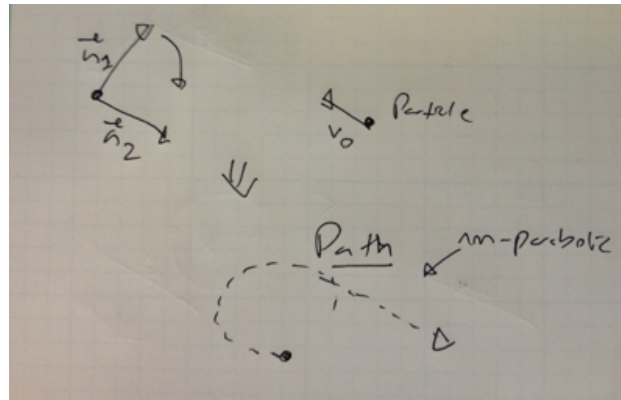


Figure 3.7: Demonstration of the particle path during concurrent rotating and free-fall

## Elastic Collisions

Collisions are now elastic, meaning particles maintain a perpendicular velocity when colliding with an edge.

### Planar Collision

When colliding simply on an edge, they bounce and transition back into freefall.

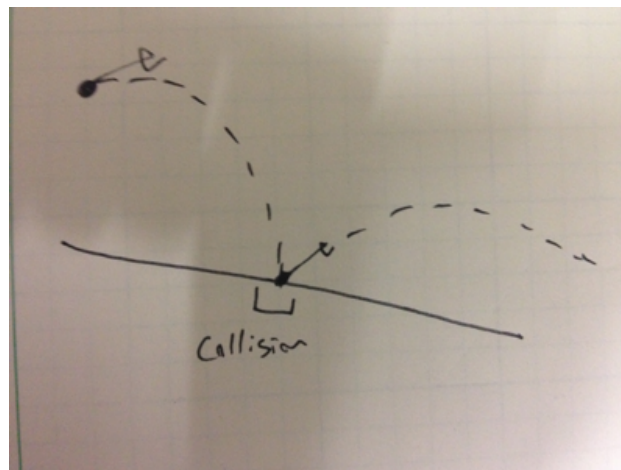


Figure 3.8: 3in

### Planar To Sliding Transition

When the particles have some  $\epsilon$  perpendicular velocity, they transition from freefall to sliding along an edge.

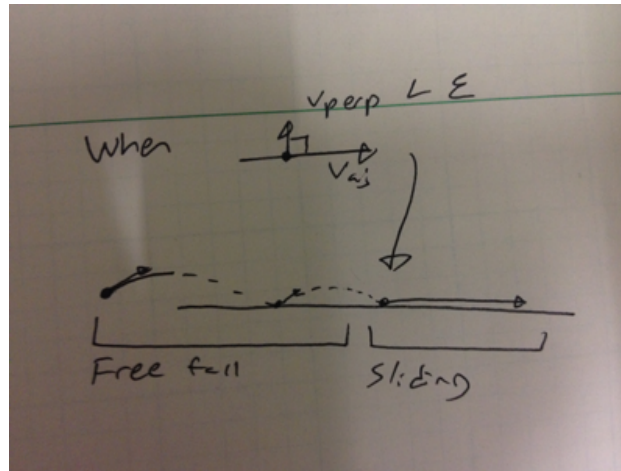


Figure 3.9: 3in

### Sliding-Edge Collision

When sliding along an edge, the particle may encounter another edge. If this edge has a dot product greater than zero, it collides with the edge and enters freefall again.

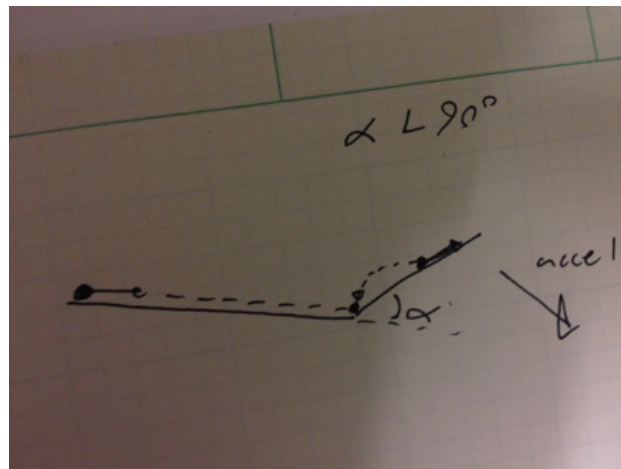


Figure 3.10: 2.5in

### Sliding-Corner Collision

If the next edge has a dot product less than or equal to 0, the particle is effectively “trapped” as long as the gravity vector points within the edge.

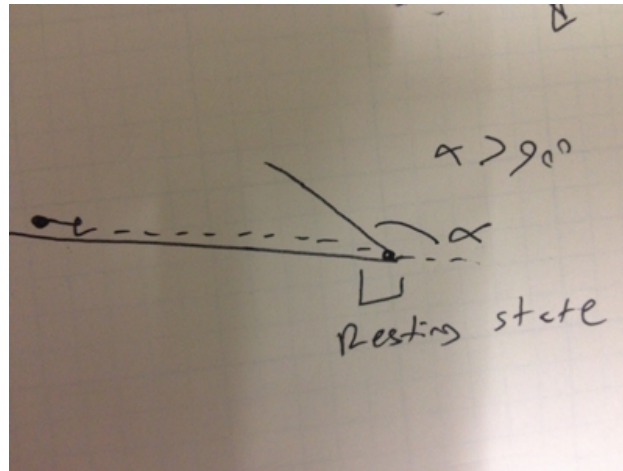


Figure 3.11: 2.5in

## Conservation of Momentum

The only source of energy in this demo is the potential energy from gravity. Note that as the workpiece rotates, the gravity vector (and corresponding field) changes. This means that a particle in a low-energy configuration can transition to a high-energy configuration.

## Settling Guarantee

The only addition of energy is from rotation, so in the absence of rotation, no energy is added to the system. Because the elasticity  $\kappa$  of the system is less than 1, energy dissipates as the particles are simulated throughout the workpiece.

## Duration of Simulation

Because of this energy dissipation, the simulation is guaranteed to terminate in one of two ways.

### Simulation End – Concave Vertex

Either the particle settles into a concave vertex with a kinetic energy less than  $\epsilon$ , or

### Simulation End – Workpiece Exit

The particle exits the workpiece, which is the goal of the simulation.

## 3.6 Results

### Run Time

It was fast, here are some numbers

### Accuracy Comparison

It was accurate in finding intersections that some euler integration schemes would not.

### With Euler Integration

## 3.7 Future Work & Discussion

### Bounding Box Method Adaption

You could modify normal bounding box methods to use parametric equations instead.

### Bounded Simulation Between Limits

There is a possibility that you could simulate between two arbitrary limits and “sweep” across the range of kinetic possibilities. Would eliminate the sampling we will see next chapter.

# Chapter 4

## Solution Search

### 4.1 General A.I. Search

General A.I. Search essentially expands from a start state and proceeds towards a goal.

#### State Space Formulation

The state space is the tuple of information that directly encodes the time-variant properties that describe your progress towards the goal. For pacman, the example is his position and the food.

Other parts of the problem that are not time-variant are considered part of the environment (for instance, the walls of pacman or the polygons in our workpiece here).

#### State Space Size

The size of the state space is all possible valid combinations of the components of the state space. E.G. for pacman, it's  $MN$  for an  $M \times N$  sized board.

#### State Space Exploration

Exploration of a state space happens through a transition function that generates successors from a state from a list of available actions.

For pacman, this means that North/South/East/West produce different states. Transition functions can fail to return a state from an action if that action is invalid.



## 4.2 Adaption of A.I. Search

Our A.I. Search algorithms will be quite similar, but our state space approach is quite different.

### Traditional Formulation

The traditional formulation for kinematic problems is to define the state of a particle as it's position, velocity, and acceleration. Successor function integrates in time.

### Our State Space Formulation

Our state space formulation is instead either:

1. The concave vertex the particle is resting in
2. The particle is out of the workpiece

Our state space is thus dramatically reduced in complexity compared to the traditional formulation, meaning our state space size will be reduced dramatically as well.

### Exploration

Instead, the bulk of the algorithm's work will be in the transition function, allowing us to determine what concave vertices we can reach from a given state.

## 4.3 Transition Function

### Definition

Transition function returns states.

### Sampling

We will sample from our available gravity directions and simulate a kinetic path from there.

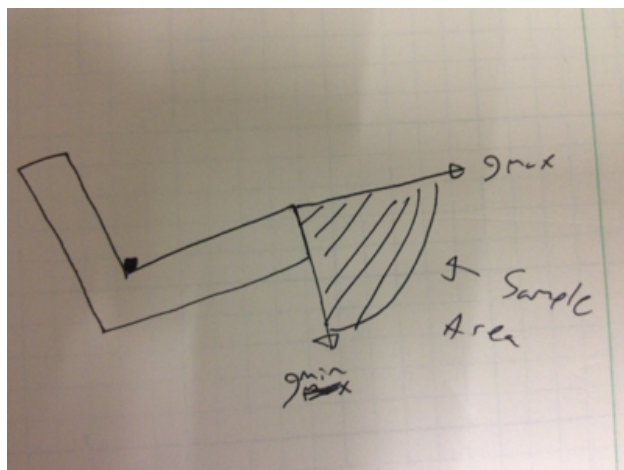


Figure 4.1: 2.5in

## Limits of Sampling

We need a gravity direction that will induce movement away from the concave vertex – this is the min. We then increase that maximum angle until the gravity direction almost causes the particle to leave the surface – we call this the maximum.

## Representative Coverage Between Limits

Try to get an even coverage over the space, sample with  $\sqrt[3]{x}$

## Graph Search

Reduce possibilities, prioritize by our “cost” for the algorithm.

### Cost Sensitive Closed List

Essentially similar to CSCL because it is order-independent.

## 4.4 Search

Now we can search from a start state outwards

### Uniform Cost Search

Using uniform cost, we always maintain optimality based on our cost.

## **Cost Function**

Our cost function can be a variety of things based on the backwards path.

### **Time**

Most popular is time, because time is money.

### **Energy - Rotation Angle**

Second most popular is energy based on the rotation angle. This would be appropriate if the workpiece was well balanced but fixture generated a lot of friction while rotating.

### **Energy - Workpiece Center of Gravity**

Another option is based on center of gravity if the energy from the workpiece rotations outweighed the friction in the motors.

## **Solution**

Defined as the sequence of gravity transitions from each concave vertex to produce a particle path that exits the workpiece.

## **4.5 Control Sequence Generator**

Now that we have a solution, we need to generate a control sequence for the workpiece rotator.

### **Sample-defined Rotations**

These rotations are defined for us by the transition function.

### **Intermediary Rotations**

Between these samples we don't have anything. What we do is just interpolate between the last sample and the beginning of the next sample with a cubic bezier curve.

## **4.6 Results**

### **Run Time**

Runs fast.

## **Part Complexity**

Transition function is constant in best case, linear in worst case based on the number of polygon edges.

## **Demonstration of Solution**

Go to this web address!

## **4.7 Future Work & Discussion**

Could be improvements

## **Heuristics for A\* Search**

A\* search can be added with heuristics. Distance to workpiece bounding box might be a good one, but that's essentially a radially symmetric heuristic, not very informative.

## Chapter 5

# Conclusion

This is the conclusion, it concludes the paper.

# Bibliography

- [1] James Moorer. “Signal Processing Aspects of Computer Music—A Survey”. In: *Computer Music Journal* 1.1 (1977).