This does not mean the code is debugged!

**Test Plans**

**Test Driven Development I**
- code test
  - write tests for new code
  - write new code
  - test new code

**Test Driven Development II**
- test new code
  - write integration test
  - integrate new code
  - test integration

**Unit Test for Vertex Collision**

- You know answer ahead of time
- Tests include problem cases
Step 1

- choose \( v \) and \( p_0 \) randomly
  1. distance < \( r \)
  2. distance = \( r \)
  3. distance > \( r \)

Step 2

- choose random vector unit \( w \)
- compute path endpoints \( v+(r-\delta)w \), \( v+rw \), and \( v+(r+\delta)w \)

Step 3

- throw out test case if dot product is positive.

Step 4

- \( 0 < \alpha < 1 \)
- \( \alpha = 1 \)
- \( \alpha \approx 0 \)

Step 5

- One jump positive tests
- Two jump positive tests
- Two jump negative tests
Questions

• Are these good tests?
• Are there other kinds of tests we should do?
• How can we automate testing?
  – Test generation
  – Running the tests

Triangle World II

1. Extend edge and face collisions to general case
2. Integrate vertex, edge, and face collisions
3. Handle up to 3 triangles
   (this is a change from overview)
4. Test plan

Devise test plans for standardized edge and face tests.