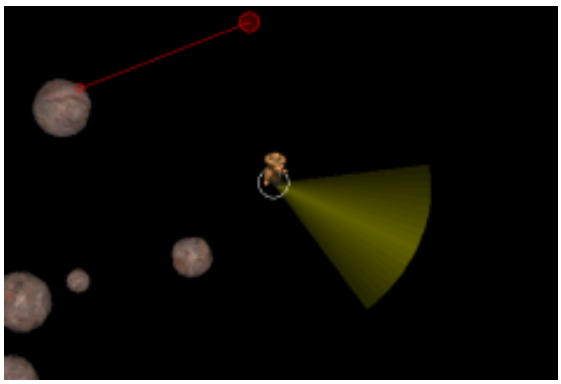
ETGG1803	Lab5: **"Boulder-Dodge" [Dot Product (+ more physics)]**	Points[1, 2]: **47**
Assigned:	**2/23/2015 (section 02: MW 9-11:15am)** **2/25/2015 (section 01: MW 12-2:15pm)** **2/26/2015 (section 03: TR 12-2:15pm)**	
Due[3]:	**Friday 3/6/2015 by 5:00pm** (right before spring break)	

Objectives:

- Work on another pygame / math3d integration project. You should, by the end of this lab, be more comfortable using 3d math in your 2d programs.
- Explore some of the many applications of dot product in graphics / games.

Tasks:

- (optional) Use your physics class from Lab3 (or mine from the solution). This *should* save you some time.
- Your program must include the following (you'll lose points from the total if these aren't present):
  - (**5 points**) Good structure: use of classes, docstrings, comments (where appropriate).
  - (**2 points**) Add a dot method to math3d. Make sure to do appropriate type-checking and documentation.
  - (**2 points**) Use the "opposing-force" method of doing friction and use dot-product to detect direction changes.
  - (**7 points**) Make a player that points towards the mouse at all times. When left-clicking, the player should accelerate in the facing direction. There should be friction to slow the player down. Drawing a circle is sufficient, but include a line indicating direction.
  - (**7 points**) Make several randomly-size, randomly-placed boulders which move in random directions [optionally include friction here too]. Use the "basic_boulder.png" and rotate and size it to face in the direction the boulder is moving.
  - (**9 points**) Make a rotating laser that, if it hits a boulder imparts a force at the intersection point. We'll go through the symbolic math for this in class.
  - (**2 points**) Subject the boulders to a "terminal velocity" of n pixels / s (you can decide what n is – make sure it's a scalar, though).
- Your program should include some of the following math-related features. You need to complete enough of these to get 100%. If you do all of them (correctly), you can earn bonus.
  - (**12 points**) Implement 2d inelastic collisions between boulders *without* using trig [it's slower than the vector-based method]. We can go through *some* of the math in class, but here's a good reference: http://www.imada.sdu.dk/~rolf/Edu/DM815/E10/2dcollisions.pdf
  - (**7 points**) Make sure nothing spawns on top of something else (laser, boulders, player). The spawn positions should be completely random (i.e. no pre-defined spots) – look for existing "things" on that spot.
  - (**8 points**) The player can "shout" in a cone-shaped blast. Anything in that blast is pushed away from the player. The simple (but probably good) solution to this is to see if any part of the boulder is within the cone. If so, impart a force away from the player – this force should be greatest at the center and weaker at the edges. Show the blast cone when it is active (an image or pygame drawing commands). Make sure, though, that it is easy (in code) to adjust both the range and angle of the cone.
  - (**9 points**) Use the "boulder.png" image and make it look like the boulder is spinning. To get full points here, you need to consider the boulder's radius *and* current speed.
- Your program can optionally contain the following non-math-related features for a few bonus points. These will probably take you some time to complete – I'd recommend focusing on the math-related portions first (since you'll likely see these on later quizzes). I'm giving slightly less points than the time / effort needed to implement these (since they don't really contain any math content).
  - (**3 points**) player HP (decreases when hit by a laser / boulder) and stamina bar (empties upon shout, refills gradually). Also, add player invulnerability when they get hit by a boulder / laser.
  - (**2 points**) Use the included Indiana Jones image (or another similar sprite-sheet with animation), including animation and pointing in the general direction the player is facing.
  - (**4 points**) Add pickups, timers, etc. to give the user a goal.
- Here's a YouTube video (the example program includes some of the optional bonus features and all the orange and red points): `http://youtu.be/sOxzMeeZtv4`.



[1] You can earn up to 70 points on this lab if you implement everything.
[2] I'm not going to use a unit-tester for this, but I am going to *primarily* test it by running the program.
[3] The due date's a bit longer than normal – don't use that as an excuse to procrastinate. **This lab is a bit harder than most**!