Note: This is (one of) last year's final exams – the topics last year might've been slightly different than this year's…

Name: _____ Score: ___ / 100

There are 110 points possible on this test. Good luck! Please turn in your cheat sheet with this test.

1. Compute the following numerically using the quantities given:

$$\vec{v} = [1 \quad -2 \quad 3] \qquad \vec{w} = [0 \quad 4 \quad 5] \qquad A = \begin{bmatrix} 0.5 & 2 & 1 \\ 1 & 3 & -1 \end{bmatrix}$$

$$B = \begin{bmatrix} 4 & 1 & 6 & 0 \\ 0 & 5 & 7 & 2 \\ 3 & -3 & 2 & 0 \end{bmatrix}$$

a. (**3 points**) $\vec{v} \bullet \vec{w}$

7.0

b. (**3 points**) $\vec{v}\, x\, \vec{w}$

$[-22 \quad -5 \quad 4]$

c. (**3 points**) $\vec{v} - \vec{w}$

$[1 \quad -6 \quad -2]$

d. (**3 points**) $\vec{v} * \vec{w}$

Undefined!

e. (**4 points**) $\vec{w} * A^T$

$[13 \quad 7]$

f. (**4 points**) $A * B$

$\begin{bmatrix} 5 & 7.5 & 19 & 4 \\ 1 & 19 & 25 & 6 \end{bmatrix}$

2. (**12 points**) Write a python class called **Asteroid** which:
   a. Has these attributes:
      i. **mVel**: a 2d math3d vector.
      ii. **mPos**: A 2d math3d object.
      iii. **mRadius**: the radius. The user should pass this as well when creating an Asteroid.
      iv. [Anything else you deem necessary]
   b. Has these methods / behaviors:
      i. To create an asteroid (these numbers are just examples – make sure your code will work with any values):

```
x = 400
y = 300
degrees = 23
radius = 20
speed = 150        # px / s
a = Asteroid(x, y, degrees, speed, radius, "brown_rock.png")
```

      ii. Make this happen (the values are position, radius, and velocity:

```
print(a)    # {<Vector2: 400, 300> 30 <Vector2: 0.0, 0.0>
```

      iii. Make this happen (i.e. determine [using circle-circle hit check] if two Asteroids overlap:

```
if a == b:
    print("Hit!")
```

      iv. Make this happen (i.e. return the degree (0 degrees = right, 90 = up, etc.) heading of the Asteroid

```
print(a.getDirection())                # 37.2
```

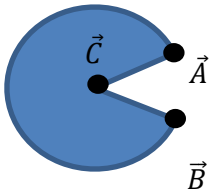
```
class Asteroid(object):
    def __init__(self, x, y, deg, spd, rad, fname):
        self.mPos = vector.Vector2(x, y)
        ang = math.radians(deg)
        self.mVel = vector.Vector2(spd * math.cos(ang), -spd
                                   -spd * math.sin(ang))
        Self.mRadius = rad

    def __str__(self):
        s = "{" + str(self.mPos) + " " + str(self.mRadius)
        s += " " + str(self.mVel)
        return s

    def __eq__(self, other):
        if isinstance(other, Asteroid):      # +2
            d = (self.mPos - other.mPos).magnitude()
            return d < self.mRadius + other.mRadius
        return False


    def getDirection(self):
        return math.degrees(math.atan2(-self.mVel[1], self.mVel[0]))
```

3. (**10 points**) Suppose you are given



   a. $\vec{A}, \vec{B}, and\ \vec{C}$: 3 points that define pacman's mouth (they're 3d but all lie on a common plane -- don't assume *what* plane it's on)
   b. $\vec{P}$: the position of a power pellet (this also lies on the plane)
   c. **n**: the radius of the power pellet.
   d. **s**: pacman's radius

   Write an algorithm to determine if any part of pacman is touching the power pellet.

My approach is to see first if P lies within the circle. Then a create two planes (one for each side of the mouth) and see if the point lies on the "front" of both. I'm sure there are other approaches to this too.

if $\|\vec{P} - \vec{C}\|$ < s + n:

$\overrightarrow{Axis} = (\vec{B} - \vec{C})\ x\ (\vec{A} - \vec{C})$ the normal to the plane containing A, B, C, and P

$\overrightarrow{N\_bot} = \overrightarrow{Axis}\ x\ (\vec{B} - \vec{C})$

$\widehat{N\_bot} = \overrightarrow{N\_bot} / \|\overrightarrow{N\_bot}\|$

d_bot = $\vec{B} \bullet \widehat{N\_bot}$

$\overrightarrow{N\_top} = \overrightarrow{Axis}\ x\ (\vec{C} - \vec{A})$

$\widehat{N\_top} = \overrightarrow{N\_top} / \|\overrightarrow{N\_top}\|$

d_top = $\vec{B} \bullet \widehat{N\_bot}$

if $\vec{P} \bullet \widehat{N\_bot} > d_{bot} + n\ and\ \vec{P} \bullet \widehat{N\_top} > d_{top} + n$:

touching = True

else:

touching = False

else:

touching = False

4. (**8 points**) Suppose you are given:
   a. $\widehat{camX}$
   b. $\overrightarrow{camPos}$
   c. $\overrightarrow{camCOI}$

   Symbolically calculate the other two camera axes in a left-handed system.

$\overrightarrow{camZ} = \overrightarrow{camCOI} - \overrightarrow{camPos}$

$\widehat{camZ} = \overrightarrow{camZ} / \|\overrightarrow{camZ}\|$

$\widehat{camY} = \widehat{camZ}\ x\ \widehat{camX}$ *No need to normalize – the other two are perpendicular.*

5. (**8 points**) Suppose you are ray and a plane. Suppose you have done a ray-plane intersection and have determined that the ray hits the plane at a distance of t. Symbolically show how to find a point which is 10 units away from the intersection point (but still along the ray – you can assume the ray is at least 10 units from the intersection).

$$\vec{P} = \overrightarrow{Ray_{Origin}} + (t + 10) * \widehat{Ray_{dir}}$$

6. (**10 points**) Show the specular part of the standard lighting equation. In your answer, describe the variable names you're using (less than one sentence each)

$\vec{P}$: *point being illuminated*
$\hat{N}$: *normal at point being illuminated*
n: *number of lights*
$\vec{L_i}$: *position of light i*
$\vec{S_i}$: *specular color of light i*
$\vec{S_m}$: *specular color of material at point P*
$\vec{C}$: *camera position*
h: *hardness of specular reflection at point P*

$\vec{Scol} = [0 \quad 0 \quad 0]$
$\hat{V} = (\vec{C} - \vec{P})/\|\vec{C} - \vec{P}\|$
For i in range(0, n):
$\hat{LD} = (\vec{L_i} - \vec{P})/\|\vec{L_i} - \vec{P}\|$
$\hat{R} = 2 * (\hat{LD} \bullet \hat{N}) * \hat{N} - \hat{LD}$
$sStr = \hat{R} \bullet \hat{V}$
If sStr > 0:
$\vec{Scol} += sStr^h * (\vec{S_m} \otimes \vec{S_i})$

7. (**8 points**) Show how to create a <u>single</u> matrix which scales by (5x, 3y, 0.5z) *then* translates by (+6x, -4y, 0z) *then* rotates by 15 degrees around the z-axis. I do want numbers in this answer, but I only care about how you set up the construction of this matrix, not the actual final numbers.

Note: you probably should've asked on this final (or assumed, with a note like this) that this is in a particular coordinate system. Here, I'm assuming it's in a left-handed system.

$$C = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 6 & -4 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos(15) & \sin(15) & 0 & 0 \\ -\sin(15) & -\cos(15) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

8. (**5 points**) When multiplying a 4x4 matrix and a 4x4 matrix, how many scalar multiplications are performed?
64 (16 dot product operations, where each dot product contains 4 multiplies)

9. (**8 points**) Merida fires an arrow from position $\vec{P}$ in direction $\hat{D}$ at a target with position $\vec{T}$ (this point represents the center of the bullseye) and normal $\hat{N}$ (this is the perpendicular to the plane the front of the bullseye lies upon). Ignore gravity, wind resistance, etc. and assume that all positions are measure in inches from some reference position (which isn't important). The target has 4 concentric rings that are each 2 inches across and are worth 100, 75, 50, 25 points each (100 is the bulls-eye). Write a symbolic algorithm to determine how many (if any) points she earns.
Treat the arrow as a ray and the target as a plane. See where the arrow hits the plane.
If $\hat{D} \bullet \hat{N} = 0$: points = 0 (arrow and target are perpendicular)
Else:
t = $\frac{\vec{T} \bullet \hat{N} - \vec{P} \bullet \hat{N}}{\hat{D} \bullet \hat{N}}$

if t < 0:

points = 0 (she was facing the wrong way!)

Else:

$\vec{H} = \vec{P} + t\hat{D}$

Dist = $\|\vec{H} - \vec{T}\|$

If Dist < 2: points = 100

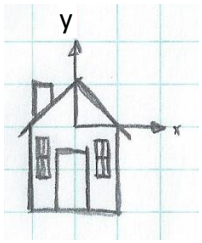
Elif Dist < 4: points = 75

Elif Dist < 6: points = 50

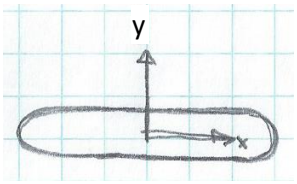
Else Dist < 8: points = 25

Else: points = 0 (+4 if you did this with a formula instead of if's)

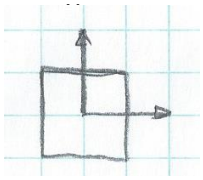
10. (**11 points**) Suppose you are given the following models (one grid mark = ½ world unit) and the scene graph shown to the right. Sketch the world *and* indicate the local coordinate systems of each node (assume left-handed world).



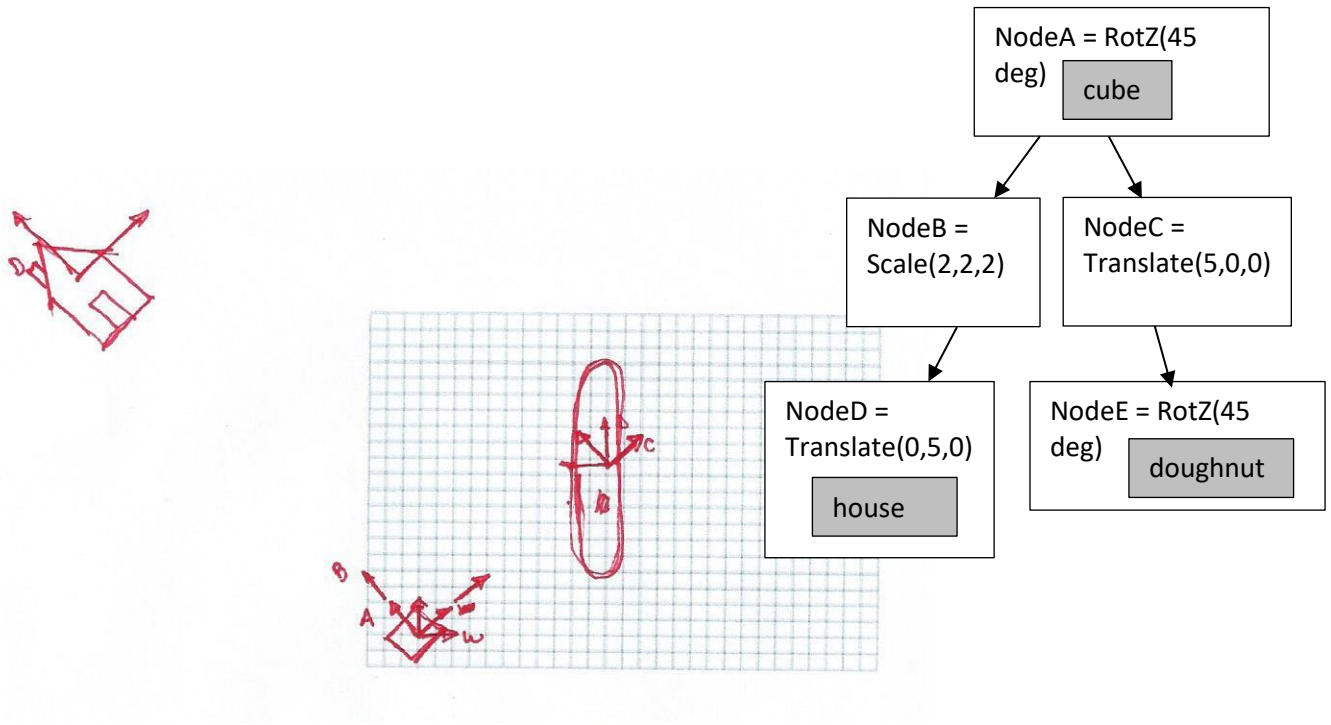
house



doughnut



cube



11. (**3 points**) I think I'll get a _Meh_ in this class [no wrong answer – just seeing if it matches reality]

Have a great summer!!