Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page.

Name: _____

Review

1. In this section, you will quickly review some of the basic concepts that will be needed for today's exercises.
   (a) In Python, what is `3 / 2`?

   (b) In Python, what is `3.0 / 2.0`?

2. Perform the following Numerical Python computations:
   (a) Import the Numerical Python module as `np`:

   (b) Compute the result of `position` and `velocity` after performing the following operations in Python:

```
position += velocity * time
velocity += acceleration * time
```

   when initially:

```
position = np.array([0,0])
velocity = np.array([0,1])
acceleration = np.array([1, 1])
time = 0.2
```

(c) Redo the above calculation with `position = np.array([0.0,0.0])`, `velocity = np.array([0.0,1.0])`, `acceleration = np.array([1.0, 1.0])`, and `time = 0.2`

(d) Why does this happen?[1]

(e) Finally, given two arrays `a = np.array([0,5])` and `b = np.array([0,5]`, indicate the Python code to determine that these are equal.[2]

Vector

3. We will now perform a linear algebra review.

(a) Assume that the boy is at $(0, 5)$ and that the girl is at $(3, 2)$. Since the girl is the AI, compute the vector $\vec{t}$ from the girl to the boy.

(b) Compute the magnitude of $\vec{t}$, that is, $|\vec{v}|$.

(c) Normalize this vector.

[1] `http://docs.scipy.org/doc/numpy/reference/generated/numpy.array.html` indicates the `dtype` initializer as one possible solution.

[2] `http://docs.scipy.org/doc/numpy/reference/routines.logic.html`

(d) Assume that the time that has elapsed since the last frame is 0.2 seconds and that the avatar has a speed of 200 pixels per second. What is the final position of the girl avatar if the simplified kinematic equation is:

$$\vec{p} \leftarrow \vec{p_o} + \vec{v}t$$

Avatar

4. For this module, you will create an `Avatar` class using the file `avatar.py`, which will use the simplified version of the Physics model that has just been discussed. That is, we will use an instantaneous acceleration model with constant velocity.

   (a) The avatar should have an initializer with the following parameters: `name`, `surface`, `position`, and `speed`. As with the lab exercise, internally convert `position` to a `numpy` array.

   (b) Add a method called `update(self, target, time)` to the Avatar class. This method will use the following calculation to update its position:

$$\vec{p} \leftarrow \vec{p_o} + \vec{v}t \tag{1}$$

   (c) Test your function (perhaps using the `__name__ == '__main__'` idiom) using `target = np.array([1.0, 5.0]` as well as `target = np.array([0.0, 0.0])`. Correct any resulting bugs (such as division by zero errors).

Game

5. For this module, you will use the provided `game_key2.py` solutions as a starting point. Save this file as `game.py`.

   (a) Remove the `girl` avatar (we will add it back later). We will also no longer worry about scrolling past the edges of the screen (so no need to use modulus).

   (b) Currently, the game is reliant on variable frame rates. This is problematic because the speed of the avatar is dependent of the frame rate. Consequently, fix the game so that it is time-based (rather than frame-based) and use `clock.tick` to restrict the game to 30 fps.

(c) Create a variable `time_passed_seconds` to reflect the amount of time that has passed since the last frame. This should be part of the game loop.

(d) Refactor the assignment so that `boy` uses numerical Python, rather than tuples, lists or anything else.

(e) Furthermore, convert `boy` so that it uses the avatar class in file `avatar.py`: `from avatar import *`. Give the `boy` a starting location of $(50.0, 50.0)$ with a speed of 200.0.

6. Now we will add the `girl`, also as an avatar. Add the `girl` to location $(300.0, 300.0)$ with a speed of 50.0. Ensure that the AI character appears on the screen (though it won't move).

7. Add a function `executeAIBehavior(enemy, player, time_passed_seconds)` to your `game.py` class. This AI function will execute the seek algorithm as discussed in Chapter 3 of the text. Don't forget to call `enemy.update` at the end of the function!

   Congratulations, you have just implemented your first AI movement algorithm – seek!