1. A high-level description of each of your public member functions in each of

your classes, and why you chose to define each member function in its host

class; also explain why (or why not) you decided to make each function

virtual or pure virtual. For example, “I chose to define a pure virtual version

of the blah() function in my base class because all actors in Space Inflators

must have a blah function, and each type of actor defines their own special

version of it.”

2. A list of all functionality that you failed to finish as well as known bugs in

your classes, e.g. “I wasn’t able to implement shooting of Septic Bullets.” or

“My Wealthy Nachling doesn’t work correctly yet so I just treat it like a

Nachling right now.”

3. A list of other design decisions and assumptions you made, e.g.:

i. It was ambiguous what to do in situation X, and this is what I decided

to do.

4. A description of how you tested each of your classes (1-2 paragraphs per

class)

1. class Actor

Actor(StudentWorld\* ptr, int id, int x, int y)

This function constructs an Actor. It sets visible to true for graphobject and in its member initialization list constructs the GraphObject.

Virtual ~Actor()

This function is the destructor. I made it virtual because I needed to because all the destructors must be called.

Virtual void doSomething() = 0;

I declared this function pure virtual because there is no plain Actor object. Each derived class has a doSomething() function.

Bool isStillAlive();

I made this function so all actors could call it in order to tell if its alive or not. They need to know if they’re alive or not in order to know if they should do something or be deleted from the data structures.

StudentWorld\* getWorld();

Many times actor classes need to call functions from studentWorld and this allows them to do that by returning a pointer to the world.

Void setAlive(bool m)

Each actor has a private member variable bool m\_alive and this function lets you set it to either true or false. This is used in many funcitons to set actors to dead when they die so that they will be deleted.

Class Ship : public Actor

I made ship a derived class of Actor because everything has to be a derived class of actor in some way.

Ship(StudentWolrd\* ptr, int id, int x, int y, int startEnergy)

The constructor takes the psotion and start energy because each ship, whether player or alien has hp. It also takes a pointer to the world so it can be passed down to the actor class in order to allow getWorld() to return the world.

Int getEnergy() const;

Its const because it doesn’t change energy but merely returns how much energy.

Double getEnergyPct() const;

This is also const because it doesn’t change energy but merely returns the percent. It returns the percent so that it can be called by the function that updates the game status line.

Void decreaseEnergy(int amount);

All ships have energy and need to have a way to decrease it

Void restoreFullEnergy();

This function restores energy to full. This is mainly for the playership when it gets an energy goodie

Bool fire(int type, int firedBy);

This function fires projectiles. It takes the type of projectile and who it was fired by.