Mudith Mallajosyula -- ID# 404937201

------------------------------------

Project 3 Report

**1. Member Functions**

**Actor.cpp**

**--------------------------------------------**

**Actor Class:**

Actor Constructor:

This is an essential function that creates a Graph Object with a type variable to serve as the base for all the actors in the program.

virtual Actor Destructor:

I declared this as virtual because some of the derived classes have individualized behaviors on destruction.

virtual void doSomething:

I declared this as a pure virtual function because every entity in the game needs one, but an Actor object will never be instantiated.

virtual bool isAlive:

I declared this as a pure virtual function because every derived class uses it to determine if it should be deleted, but different types require different implementations.

virtual void kill:

I declared this as virtual to set an objects's state to dead because the aliens, projectiles, and goodies need it to be cleared.

virtual double getHP:

I declared this as virtual because the getHP function for the aliens and the nachemblaster needs it.

virtual int travDir:

I declared this as virtual because the spaceships and projectile types use it to check collisions, and when I iterate through the Actor\* list, it needs a discrete function at compile time.

virtual void setDir:

I declared this for similar reasons as the function above; projectiles and spaceships need it and the iterator loops require it to compile.

int getType:

I declared this as virtual because it is necessary for collision checking for every spaceship, projectile, and goodie.

**Star Class:**

Star constructor:

This is an essential function to construct star objects.

void doSomething:

This function controls the star's leftward movement. Not virtual because Star has no derived classes

bool isAlive:

This function checks if the star is in the bounds. Not virtual because Star has no derived classes

**Explosion Class:**

Explosion constructor:

This is an essential function to construct Explosion objects.

void doSomething:

This function controls the explosion's movement and its lifespan

bool isAlive:

This function controls the explosion's lifespan

**Spaceship Class:**

Spaceship Constructor:

This is an essential function to create spaceship-type objects

virtual Spaceship Destructor:

This is declared as virtual because aliens have distinct behavior on destruction.

double getHP:

This funciton returns the HP of the spaceship

double getSpeed:

This funciton returns the speed of the spaceship

void setSpeed:

This funciton allows you to set the speed of a spaceship

void damage:

This funciton allows you to damage a spaceship, takes the type of the object doing the damage as an argument

void checkCollision:

This funciton checks collisions between actors. Only the spaceships do this.

StudentWorld\* getWorld:

This funciton returns a pointer to the StudentWorld the spaceship is in

virtual bool isAlive:

This checks if the spaceship is alive. It is declared as virtual because NachenBlaster and Alien types have different requisites for being alive.

**NachenBlaster Class:**

NachenBlaster constructor:

This is an essential function that constructs the NachenBlaster object.

void doSomething:

This function interprets user input, controls movement, and checks for collisions of the NachenBlaster

void incTorpedoes:

This function increments the number of torpedoes the NachenBlaster has.

int getTorpedoes:

This function returns the number of torpedoes the NachenBlaster has.

int getCabbage:

This function returns the amount of cabbage ammo the NachenBlaster has.

**Alien Class:**

Alien constructor:

This is an essential function that constructs alien type objects.

Increments the number of ships in the world.

Alien Destructor:

This function decrements the number of ships in the world.

void setDeltaX:

This function changes the delta x of the alien

void setDeltaY:

This function changes the delta y of the alien.

void setFlightLen:

This function changes the flight len of the alien.

int getDeltaX:

This function returns the current delta x of the alien.

int getDeltaY:

This function returns the current delta Y of the alien.

void move:

This function controls the movement of the aliens

bool fire:

This function decides whether or not to fire a projectile.

virtual void checkFlightPlan:

This function checks the current flight plan and updates it as necessary. I declared this as virtual because the snagglegon follows a different path.

**Smallgon Class:**

Smallgon Constructor:

This is an essential function for creating smallgon objects

void doSomething:

This function controls the movement of the smallgon, checks and modifies its flight plan, decides whether or not to fire on the NachenBlaster, etc.

**Smoregon Class:**

Smoregon Constructor:

This is an essential function for creating smoregon objects

void doSomething:

This function controls the movement of the smoregon, checks and modifies its flight plan, decides whether or not to fire on the NachenBlaster, etc.

**Snagglegon Class:**

Snagglegon Constructor:

This is an essential function for creating Snagglegon objects

void doSomething:

This function controls the movement of the Snagglegon, checks and modifies its flight plan, decides whether or not to fire on the NachenBlaster, etc.

void checkFlightPlan:

This function controls the movement and updates the direction of the snagglegon as necessary

**Projectile Class:**

Projectile Constructor:

This is an essential function for creating Projectile objects

bool isAlive:

This function checks that the projectile is within the bounds and alive

void kill:

This function sets the projectile's status to dead.

int travdir:

This function returns the direction the projectile is traveling.

void setDir:

This function sets the travel direction of the projectile

**Cabbage Class:**

Cabbage Constructor:

This is an essential function for creating Cabbage objects

void doSomething:

This function controls the movement and rotation of the cabbage

**Turnip Class:**

Turnip Constructor:

This is an essential function for creating Turnip objects

void doSomething:

This function controls the movement and rotation of the turnip

**Torpedo Class:**

Torpedo Constructor:

This is an essential function for creating Torpedo objects

void doSomething:

This function controls the movement and rotation of the torpedo

**Goodie Class:**

Goodie Constructor:

This is an essential function for creating Goodie objects

pure virtual Goodie desturctor:

This was implemented as pure virtual so that no goodie objects could be instantiated.

void doSomething:

This function controls the movement and rotation of the goodie

**ExtraLifeGoodie Class:**

ExtraLifeGoodie Constructor:

This is an essential function for creating ExtraLifeGoodie objects

**RepairGoodie Class:**

RepairGoodie Constructor:

This is an essential function for creating RepairGoodie objects

**TorpedoGoodie Class:**

TorpedoGoodie Constructor:

This is an essential function for creating TorpedoGoodie objects

double euclidean\_dist:

This function returns the euclidean distance between two objects

**StudentWorld.cpp**

-------------------------------------------------------

**StudentWorld Class:**

StudentWorld Constructor:

This is an essential function for creating StudentWorld objects

StudentWorld Destructor:

This function calls the cleanup function.

int init:

This function sets up the game whenever it starts or resets

int move:

This function checks if objects are dead and calls the doSomething function for each actor

void cleanUp:

This function deletes all the actors in the world

void addActor:

This function adds an actor to the actor array of the world

void decShips:

This function decrements the number of ships in the world

void incShips:

This function increments the number of ships in the world

void incKills

This function increments the number of kills in the level

NachenBlaster\* getPlayer:

This function returns the player of the world

list<Actor\*>\* getActors:

This function retuns a pointer to the actor list

2. **Failed to Finish**

I think I implemented all the functionality, but I may have missed something minor. I was unable to test the sounds since I was running on Linux, so I may have left some bugs unchecked.

One of the bugs I noticed is that it seems that sometimes the things that rely on random number generation (eg creating ships, choosing when to fire a projectile) tend to repeat more than seems probable by chance. Sometimes 5-6 of the same type of ship would be created at the same time.

Also, even though it should have a 1/6 probability of showing up, it seems as though it is very rare for a torpedo goodie to drop.

3. **Design Assumptions**

I assumed that the score would not be decremented after a death, that Smoregons still shoot after they start their charge, and that a GraphObject could move a decimal number with the moveTo function.

4. **Tests**

**Actor.cpp**

-------------------------------------------

**Actor Class:**

I tested that the actor class could not be created.

**Star Class:**

I tested that stars could be created, and observed visually that the doSomething function worked as intended.

**Explosion Class:**

I tested that explosions could be created, and observed visually that the doSomething function worked as intended

**Spaceship Class:**

To test the spaceship class, I made sure that a spaceship could not be created. Each of the member functions of the spaceship class were tested on the derived classes.

**NachenBlaster Class:**

To test the NachenBlaster, I first tested that it could be created, that its member variables were initialized correctly, and that the directional controls worked as intended. I then tested that the torpedoes would not fire when none were present and that the would when the blaster had torpedo ammo. I further tested it on collision with each type of alien and each type of projectile (including cabbage). I then tested collisions with each type of goodie and

made sure they acted as was required. I checked that each of the member functions of Spaceship could be applied to NachenBlaster, using both correct nad incorrect input data for the arguments of setSpeed, checkCollisions, and damage functions.

After this, I did a visual test to make sure the NachenBlaster could not leave the bounds of the screen and that it interacted correctly with the environment.

**Alien Class:**

To test the Alien class, I made sure that an object of type Alien could not be instantiated, and then checked all of its member functions with the Smallgon, Smoregon, and Snagglegon types.

**Smallgon Class:**

I first tested that a smallgon could be created, and that the doSomething made it move as required (on visual inspection, it did). I then checked that the functions inherited from the Spaceship class worked as intended. I further checked the functions of the Alien class, first checking them with valid and then invalid inputs. I checked the destructor behavior of the class and tested the probability that it would shoot when in line with the Nachenblaster. I also made sure that the checkFlightPlan function and the move functions exhibited correct behavior.

**Smoregon class:**

I followed much the same process as above to test this class, with the exception that I had to test the drops of goodies when the smoregon is killed, and that the flightplan was changed if the smoregon begins to "charge."

**Snagglegon Class:**

I followed pretty much the same process as above, but made sure that the snagglegon only followed diagonal paths, and that it only shot torpedoes. I also tried to check, for all the aliens, that they spawned according to the correct probabilities, but the spawn rate seemed to be rather inconsistent.

**Projectile Class:**

To check this class, I made sure that an object of its type could not be instantiated, and checked its member functions using the derived classes below.

**Cabbage Class:**

To check this class, I made sure that it was deleted when it left the bounds, and checked that it rotated and moved as required in the doSomething class. I also checked the isAlive, travDir, setDir and kill functions on it.

**Turnip Class:**

I followed the same process as above, making sure that the projectiles traveled in the opposite direction.

**Torpedo Class:**

I followed the same process as above, except that the projectile did not rotate. I also made sure that torpedoes traveling in the same direction as the NachenBlaster did not damage it, and that torpedoes traveling towards it did not damage any aliens.

**Goodie Class:**

To test this class, I made sure that it could not be instantiated and tested its doDomething function with the derived classes.

**ExtraLifeGoodie Class:**

I tested that this class could be instatiated and that it moved as required.

**RepairGoodie Class:**

I tested that this class could be instatiated and that it moved as required.

**TorpedoGoodie Classs:**

I tested that this class could be instatiated and that it moved as required.

**StudentWorld.cpp**

-----------------------------------------------------------------

**StudentWorld Class:**

To test the StudentWorld class, I tested that an object of its type could be instantiated, that the GameWorld functions could be called on it, and that its member variables were instantiated correctly. I then tested the member functions, starting with the init function. I made sure that the variables were reset correctly and that the on-screen text was correct.

To test the move function, i created an array of actors and manually set some of them to dead, and then checked that the behavior of each was correct. I then created a normal instance of the game and checked to see that it appeared to work as required. I then tested the functions to add new actors, increment and decrement the number of ships, increment the number of kills, and clean up the world, checking that the behavior of the move and init functions was correct after this. Finally, I checked that the getPlayer function correctly returned a pointer to the NachenBlaster and that the getActors function returned a pointer to the actor list.