CS32 Project 3: NachenBlaster

1.

Actor:

The only pure virtual function was doSomething because each actor has to do something during each tick. Other than that, the public functions were mostly setter and getter functions for the private member variables that were common to all actors. I made isAlive() virtual because different actors have different definitions for it (like NachenBlaster can’t go off screen), but the base one checks if the object has gone off the side of the screen or if it’s state has been set to dead. I also have a firedByPlayer function that returns true only if that actor was fired by NachenBlaster. I defined a virtual function collision because different base classes (Alien, Projectile, Goodie) use it in different ways, but its not pure virtual because some Actors don’t use that function. I also have a getWorld() function that returns a pointer to StudentWorld. I used this function a lot to connect the two cpp files.

Damageable Object:

This class is an abstract base class because it doesn’t define doSomething. In this class I made a hitPoints() function that returns how many hit points the damageable object has left. increaseHitPoints() increases the hit points and sufferDamage lowers the hitPoints of the actor by the amt given. None of these are virtual because they are never redefined. These are used by Alien and NachenBlaster to alter the hit points they have.

NachenBlaster:

In the doSomething function, I use a StudentWorld pointer to receive input. Using a switch statement, I check which key is pressed a react accordingly (moving the NachenBlaster to whatever direction or firing a projectile). I make sure to decrement cabbage energy or number of torpedoes if firing a projectile. At the end of each tick, 1 cabbage energy is added if the energy points are less that the max. I redefined isAlive() so that it only returns false when the NachenBlasters hp is 0 or less, not if it flies off the screen because it can’t. the other functions are self-explanatory by the names. They get the percentage of hit points and cabbage energy and the number of torpedoes the player has. incFlatTor adds 5 torpedoes and is called when picking up a torpedo goodie.

Alien:

I redefined the isAlive() function to set Alien to dead when its hit point are 0 or when it flies off the left side of the screen only. I also have a function that sets a new flight plan when the aliens flight plan is zero or when it touches the top or bottom of the screen. Then I have some getter and setter functions for flight plan length and travel direction and speed. The move() function, moves the alien the a direction specified by my global constants. I also have a function that tells the user if the player is in the line of fire and what to do. Finally, there is a collision function that is virtual because other classes have it and its good style. First, collision checks if the alien has been hit by the player, if they have the alien dies and damages the player and returns true. Then collision function uses a function in StudentWorld that cycles through each actor (excluding player) and if its colliding with an actor that has been fired by NacheBlaster damages the alien, if the alien dies it explodes and the player gets points. If there is no collision, the function returns false.

Smallgon:

Smallgon only defines the doSomething function. It checks if its alive, then checks if it has collided, if it has it returns, if not it checks its flight plan and checks if the player is in its line of fire and then moves. Then it checks if it has collided again.

Smoregon:

The doSomething function checks if it’s alive, then checks if it has collided, if it has it returns, if not it checks it’s flight plan and checks if the player is in its line of fire and then moves. Then it checks if it has collided again. The dropGoodie function is virtual because Snagglegon also drops goodies. The function decides if the killed Smoregon drops and item and which one to drop. This is implemented in the destructor.

Snagglegon

The doSomething function checks if the Snagglegon is at the top or bottom of the screen and resets it travel dir. Then it checks if it has collided with anything, if not checks line of fire and if it should fire a torpedo, then it moves, and checks if it has collided again. The drop goodie function determines if it should drop an extralife goodie, also called in the destructor.

Projectile

The collision function checks if the projectile fired by an alien has hit the player or not. If it has, it damages the player and sets its state to dead and returns true. Else it returns false.

Cabbage

The doSomething function moves the cabbage and rotates it by 20 degrees. Doesn’t check for collision because the alien class is already checking.

Turnip

The doSomething function checks for collision, then moves the turnip and rotates it by 20 degrees, then checks again for collision.

Torpedo

The do Something function checks for collision, then checks who fired it and moves it according to that direction, then checks for collision again.

Star

Moves the star to the left.

Explosion

Plays explosion for 4 ticks (keeps a counter) and plays sound blast at the beginning, expanding the size each tick.

Goodie

The do something function checks for collision and check what value the goodie has and implements it on NachenBlaster. The function moveOffScreen moves the goodie offscreen to the left bottom. The collision function checks if the goodie collided with player.

Each goodie has a value() function that does something different to the NachenBlaster depending on which goodie it is.

StudentWorld

Destructor calls cleanup;

init() creates a background of stars and a new player pointer, and resets number of aliens killed and number of aliens on screen.

move() uses string streams to print out the game stats, makes player do something, then makes every actor in the list do something. Checks if every actor is alive, if not, delete, if it was an alien, decrement number of aliens on screen and increase number of aliens killed. Checks if new aliens should be added and adds them. Checks if new starts should be added. Checks if enough aliens have been killed to move onto the next level. Then checks if the player is alive, if not decrement lives and resests vales and returns player died, else continues game.

cleanup() deletes player and iterates through list of actors and deletes each actor.

collision() checks if the Euclidian distance is shorter than the radiuses.

circleThruActors() iterates through all the actors and checks the first thing it collides with that’s been fired by the player.

Then there are a bunch of functions that just create new actors. These are called by the actor class.

2. I think I finished everything, though my aliens seem to freeze when they get hit by a projectile.

3. I decided to have each actor have damage value and a score value (even though most were set to zero). I decide create a collision class in Actor and StudentWorld and made them use each other.

4. I didn’t test my actor base class.

But for part 1, I first created stars and if they looked correct and moved and disappeared according to the spec I thought it was fine.

For NachenBlaster I implemented the direction keys first and just tested in game if the ship responded to key presses. For the projectile firing part, I just temporarily gave the NachenBlaster 100 torpedoes and did the cabbage calculation. Then I tested frame by frame if they collided with alien ships and were destroyed after.

I created certain aliens and saw how they moved and how much damage they could take and cause and wat goodies they dropped.

For Projectile, I just fired a lot of cabbages and torpedos and froze the frame to see the aliens fire projectiles.

For explosions, I just watched it frame by frame.

For Goodie, you can tell using the GameStats string if it did the right thing.

Honestly I didn’t really test anything, I just ran the program while coding to see if it looked like it worked.