1. **Classes**
2. Actor class

-virtual void doSomething = 0

I chose to define a pure virtual version of the doSomething() function in Actor class because all of the derived classes much have doSomething() function of their own.

-StudentWorld \* getStudent()

Returns pointer to StudentWorld class

-in health() const

Returns health of the actor.

I made it in host class because all actors have their own hitpoints.

-virtual void addGold()

does nothing.

I made addGold() virtual because some of the derived classes need to use it. I didn’t make it pure virtual since only few classes need to define their own.

-virtual bool isGoldForProtester() const

Returns false

I made isGoldForProtester() virtual so that I can differentiate temporary golds dropped by frackman from map generated golds.

-virtual bool isAlive() const

Returns true if health is not 0 and false otherwise.

I made it virtual because isAlive() function is different for protesters.

-virtual bool canActorPassThroughMe() const

Returns true.

I made it virtual function because some actors such as oils, golds, and frackman can be passed through while dirt and boulder can’t be.

-virtual void getAnnoyed(int amt)

Subtract amt from hitpoints.

I made it virtual function because some of the derived actor classes such as frackman does specific action when getAnnoyed is called

-virtual bool canDiggThroughDirt() const

Returns false.

I made it virtual so I can differentiate frackman from other actors.

-virtual bool canPickThingsUp() const

Returns false.

I made it virtual function because most actors can’t pick things up while frackman and protesters can. To differentiate actors.

-virtual bool huntsFrackman() const

Returns false.

I made it virtual because all of the actors except protester can’t huntfrackman. I made it to differentiate protesters from other actors.

-void setDead();

Sets hitpoints to 0;

I didn’t make it virtual because this function is essentially the same for all of the actors.

1. Dirt Class

-virtual void doSomething()

Does nothing.

I have to make this function because I made it pure virtual in the base class.

-virtual bool canActorPassThroughMe() const

Returns false.

Dirt cannot be passed through but canActorPassThroughMe() function in the base class returns true, so I need to make canActorPassThroughMe() function for Dirt that returns false.

1. Boulder Class

-virtual void doSomething();

See if there is dirt right below. If there is, wait 30 ticks and then start falling. when it is falling, check to see if it can keep falling or there are any actors who can be annoyed. If It hits the dirt, it disappears and if it hits frackman or protester, annoy them with 100 points.

Because doSomething is pure virtual function, all the actors must define their own.

-virtual bool canActorPassThroughMe() const

Returns false.

canActorPassThroughMe() returns true in base class so I must define boulder’s since it cannot be passes through

-bool stableState()

Returns true if its stable.

-bool waitingState()

Returns true if its in waiting state.

-bool fallingState()

Returns true if its in falling state.

1. FrackMan class

-virtual void doSomething

It takes an input key and does what it is supposed to do when that key is inputted. For example, if space is inputted, frackman creates squirt that shoots in front of him. It uses sonarkit when z is inputted. It dies when escape is inputted. It puts down its gold nugget when tab is inputted and move with arrow keys.

-virtual bool canDigThroughDirt() const

Returns true.

canDigThroughDirt() in actor class returns false. Because Frackman can I need to make its own function that returns true.

-virtual bool canPickThingsUp() const

Returns true.

canPickThingsUp() in actor class returns false so I must make the function that returns true.

-virtual void getAnnoyed(int amt);

Subtracts amt from its hitpoint. If frackman dies, plays crying sound.

Because FrackMan has unique action when it is annoyed, I must redefine getAnnoyed() function in its class.

-void addSonar()

Adds a sonarkit to frackman.

-void addWater()

Adds a waterpool to frackman.

-int getGold() const

Returns how many gold frackman has.

-int getSonar() const

Returns how many sonarkit frackman has.

-int getWater() const

Returns how many water shots frackman has.

1. Squirt Class

-virtual void doSomething()

If there is a protester, annoy it. If it has traveled 4 spaces, kill it. If It can move in the direction squirt is in, then move one space in the direction. If it can’t move, kill it.

I need to define doSomething() function for Squirt because it is pure virtual function and it does something unique to only squirt class.

1. ActivatingObject

-virtual void doSomething() = 0

Does nothing.

I made doSomething() pure virtual in this class because all of its derived classes will have their own doSomething().

-bool actByPlayer()

Returns true if it is triggered by player.

-bool actByProtester()

Returns true if it is triggered by protesters.

1. OilBarrel Class

-virtual void doSomething()

If its nearFrackman make it visible. If its picked up by frackman, disappear and play found oil sound and give 1000 points to frackman.

I must define doSomething() function because it is pure virtual and has unique declaration for oil class.

1. GoldNugget Class

-virtual void doSomething()

If it is near frackman and it is map generated gold, show itself. If frackman picks it up, play found goodie sound and give 10 points to frackman and give frackman gold. If it is gold dropped by frackman and it is picked up by protester, play protester found gold sound and give protester gold. If it is dropped by frackman and not picked up, after 30 ticks make it disappear.

doSomething() function is pure virtual so I must define one for gold nugget.

-bool isGoldForProtester() const

Returns true if gold is dropped by frackman for protesters.

1. SonarKit Class

-virtual void doSomething()

If sonarkit is picked up by frackman, play found goodie sound and give frackman a sonarkit and 75 points. If it is not picked up within given ticks, disappear.

doSomething() function is pure virtual so I must define one for sonarkit class.

1. WaterPool Class

-virtual void doSomething()

If waterpool is picked up by frackman, play got goodie sound and give frackman water and increase score by 100. If it is not picked up within given ticks, disappear.

doSomething() function is pure virtual so I must define one for waterpool class.

1. Protester Class

-virtual void doSomething()

Reduce all of count ticks variable by 1.

doSomething() function is pure virtual function so I must define one for Protester Class

-virtual bool canPickThingsUp() const

Return true.

canPickThingsUp() function returns false in actor class. Since protesters can pick things up I must redefine this function.

-virtual void addGold() = 0

I made addGold() function pure virtual for protester class because its derived class which are regular protester and hardcore protester classes have their own addGold() functions.

-virtual bool huntsFrackman() const

Returns true.

huntsFrackman() function in actor class returns false but protesters do hunt frackman so I redefined huntsFrackman() function.

-void setMustLeaveOilField()

Set it so that protester is in leaveoilfield state and reset waiting ticks to 0 so theres no delay.

-void stun()

Add more ticks to ticks to wait so protester is waiting state for longer.

-bool resting()

Returns true if protester is in resting state.

-bool leavingState() const

Returns true if protester is leaving the field.

-bool shout()

Returns true if it has been more than 15 ticks since protester shouted at frackman.

-int numSquaresLeft() const

Return the number of squares left until protester changes direction.

-void setNewnumofSquares()

Set new number of squares protester will move in one direction.

-void setNewDirection()

Give protester new direction. If protester haven’t made perpendicular turn in 200 ticks, turn perpendicular.

-void setNumOfSquaresToZero()

Set number of squares left to 0 so protester moves without delay next tick.

-int getTicksPerp()

Return how many ticks it has been since last perpendicular turn.

-void resetPerp()

When protester makes perp turn, reset tick count.

1. RegularProtester

-virtual void doSomething()

If its dead or resting or leaving do nothing. It there’s a frackman nearby shout at him. If theres a clear line of sight to frackman turn and move one square towards him. If it has moved certain number of squares in one direction, set new direction. If it can move, move in its direction.

doSomething() is a pure virtual function so I must define one for RegularProtester since it has its own things to do when doSomething is called.

-virtual void addGold()

Play protester found gold and increase score by 25. Set protester to leaveoilfield state.

Because addGold() is pure virtual function in protester class, I must redefine it.

-virtual void getAnnoyed(int amt)

If it is in leaving state, do nothing. When it is annoyed, play protester annoyed sound and stun it. If it kills protester play protester give up sound and set it to leaveoilfield state. If it dies by squirt give 100 points to frackman. If it dies by boulder, give 500 points.

1. HardcoreProtester Class

-virtual void doSomething()

If its dead or resting or leaving do nothing. It there’s a frackman nearby shout at him. If theres a clear line of sight to frackman turn and move one square towards him. If it has moved certain number of squares in one direction, set new direction. If it can move, move in its direction.

doSomething() is a pure virtual function so I must define one for HardcoreProtester since it has its own things to do when doSomething is called.

-virtual void addGold()

Play protester found gold sound and increase score by 50 and stun the protester.

Because addGold() is pure virtual function in protester class, I must redefine it.

-virtual void getAnnoyed(int amt)

If it is in leaving state, do nothing. When it is annoyed, play protester annoyed sound and stun it. If it kills protester play protester give up sound and set it to leaveoilfield state. If it dies by squirt give 250 points to frackman. If it dies by boulder, give 500 points.

1. StudentWorld Class

-virtual int init()

Generates the world map with correct number of boulders, golds and oil barrels.

Also generates frackman and first protester. Add the actor pointers into vector STL

-virtual int move()

Writes the heading and moves all of the actors. It also randomly generates waterpool and sonarkit. It deletes all of the dead actors and checks if frackman is dead or not. If frackman is dead, end the game. If frackman finishes the game, go to next level. If frackman is still alive continue the game.

-virtual void cleanup()

Removes all of the actors and erase everything in vector STL.

-bool clearOfDirt(int x, int y) const

Returns true if 4x4 space at (x,y) position is clear of dirt

-void setDisplayText()

Write the heading.

-string format(…)

Format string for heading in a nice way.

-void foundOil()

Decrease number of oils left.

-void giveFrackManSonar()

Give frackman a sonarkit.

-void giveFrackManGold()

Give frackman a gold nugget

-void giveFrackManWater()

Give frackman a water squirt.

-void addActor(Actor\* a)

Add actor\* to vector.

-void clearDirt(int x, int y)

Remove 4x4 space of dirt at (x,y) position

-bool isDirt(int x, int y) const

Returns true if there’s dirt at (x,y) position

-void removeDirt(int x, int y)

Remove dirt at (x,y) position and play digging sound.

-bool radius(Actor\* a, int x, int y, int radius) const

Returns true if actor a is within certain radius around (x,y) position.

-bool canActorMoveTo(Actor \*a, int x, int y) const

Returns true if actor a can move to (x,y) position. Checks to see if theres anything preventing actor a from moving to (x,y) such as dirt or boulder.

-int vtrNum() const

Return the size of vector.

-Actor \* frack() const

Return pointer to frackman

-Actor \* findNearbyFrackMan(Actor\* a, int r) const

Return frackman if he is near actor a within the radius of r. If frackman is not near, return nullptr

-Actor \* findNearbyPickerUpper(Actor \* a, int r) const

Return nearby protester if he is near actor a within the radius of r. return nullprt if there is none.

-Actor \* nearFrackMan(int r) const

Return actor that is near frackman within the radius of r.

-bool lineOfSightToFrackMan(Actor \* a) const

Returns true if a has clear line of sight to frackman.

-void findEmptySpot(int &x, int&y)

Find random empty space in the oilfield.

1. **Failed functionality**
2. I failed to find the algorithm to make protesters find its way back to (60,60) using queue ADT when they die. Instead I made them just disappear when they die.
3. I failed to figure out how to make hardcore protester to detect nearby frackman using cellphone signal.
4. I failed to perfectly implement clear line of sight function. It works but not perfectly.

**3. A list of all functionality and assumptions I made**

**4. A description of testing**

1. To test frackman class, I created frackman at the right coordinates and see if it moves according to what I input using the keyboard. I would check if it goes through boulder and dirt when its not supposed it. I would check if it get damaged by boulder and protesters. I tested if I could squirt water and drop gold. I also tested if I could die by pressing escape button and using sonar kit by z button.

2. To test dirt, I check if it correctly disappears when its digged

3. To test boulder, I check if it falls when it supposed to and damage frackman and protester when it supposed to. I also checked to make sure it falls after 30 ticks and disappears when it hits dirt.

4. To test squirt, I checked to see if it generates when I press space on frackman and shoots. I also check to make sure it disappears when there’s an actor in front of it and travels 4 squares.

5. To test OilBarrel. I check to make sure it sets itself to visible and is picked up when frackman is near and correctly increases the score.

6. To check goldnugget, I check to make sure it sets itself to visible and is picked up when frackman is near. I also checked to make sure it has 2 different states where in one state, it can be picked up by frackman and in the other state, it is picked up by protester. I also check to make sure it constructs when I press tab on frackman and disappears after certain ticks has passes and is not picked up.

7. To test SonarKit, I made sure it constructs correctly in the correct coordinate and correctly gives frackman points and play the right sound and disappears when it is not picked up for certain ticks.

8. To test waterpool, I checked to make sure it gives correct number of points to frackman and play the right sound. I also checked to see if it disappears when it is not picked up for a certain amount of ticks.

9. To test RegularProtester, I check to make sure it only walks in one direction for certain ticks and sets new direction. I check to see if it turns around when it has clear sight of frackman. I also tested to see if protester correctly plays the sound when it is supposed to when it is annoyed for picks up gold or shot. I tested to see if it picks up the gold I drop and hurt frackman.

10. To test HardcoreProtetster I check to make sure it only walks in one direction for certain ticks and sets new direction. I check to see if it turns around when it has clear sight of frackman. I also tested to see if protester correctly plays the sound when it is supposed to when it is annoyed for picks up gold or shot. I tested to see if it picks up the gold I drop and hurt frackman.

I also tested to see if it gives the correct amount of points to frackman.

11. To test StudentWorld class, I check to see if it correctly initializes all the actors. I also check to see if it updates the heading everytime and all the actors move each tick. I tested if it randomly generates objects correctly and end and starts the game correctly.