PROJECT 3 REPORT

Mounika Narayanan

504608921

1. DESCRIPTION OF PUBLIC MEMBER FCTS

* Actor
* virtual void hitByBoulder() = FrackMan, Protestor, and Hardcore objects utilize this function when a boulder falls on them. In the Actor class, the function is empty, but it is redefined in the classes mentioned above (explaining why it is virtual).
* void processTicks() = this function is solely responsible for the actions of each Actor because it manages the ticks and dictates when a certain Actor calls its doSomething(). If the Actor is not alive, it returns immediately. If the sleeping ticks are greater than 0, it decrements them. Otherwise it calls the Actor’s doSomething()
* void setSleepTicks(int n) = takes an integer and sets the sleep ticks; this function is utilized by Boulder & Protestor classes to set the sleep ticks under unique circumstances
* virtual void doSomething() = every derived class from Actor must do their own action every tick. In this class the definition is empty, but in the following classes it is specified.
* virtual bool canbeBribed() const = this function is set by default to false since all objects except for Protestors cannot be bribed. However, the function is virtual since it needs to be redefined in the Protestor class to true. This function serves to differentiate Protestors from all other actors in the vector in StudentWorld.
* virtual bool canbeAnnoyed() const = this function by default returns true, but it is redefined in FrackMan and Protestor since these two actors can be annoyed
* virtual void annoy() = this function is empty in this Actor class but is redefined in Protestor and FrackMan because it is called when FrackMan squirts at a Protestor & when FrackMan is shouted at
* virtual void bribeWithGold() = this function is empty in this Actor class but is redefined in Protestor because it is called when the Protestor finds gold
* virtual bool willBlock(int x, int y, bool canDig) = this function by default returns false since most of the Actors cannot block another Actor’s path; however, it is redefined in Boulder and Dirt
* double distanceFrom(int x, int y) = returns the distance from the object to x, y
* bool isAlive() const = returns if Actor is alive or not; is used by all Actors except Dirt
* void setDead() = sets the Actor to dead; is used by all Actors except Dirt
* Dirt (derived from Actor)
* virtual bool willBlock(int x, int y, bool canDig) = derived from Actor & if the Actor cannot dig through dirt (aka is not FrackMan), it returns true because Dirt can block all other Actors
* FrackMan (derived from Actor)
* virtual bool canbeAnnoyed() = derived from Actor & is modified to return true
* void doSomething() = calls ploughDirt() & takes input from the user and by using a switch statement, decides to call stepMove(direction) and/or any other bool FM function that does the specific job if it can
* void ploughDirt() = is called in doSomething(); checks to see if FM can plough dirt via a bool private member variable, then calls clearDirt() fct in StudentWorld and plays digging sound. This fct is defined in this class bc only FM can plough dirt.
* void stepMove(Direction dir) = is called in doSomething() when the player presses a directional key. The fct changes direction FM is facing if it needs to; otherwise if FM canMove() via the StudentWorld fct then FM sets itself true for digging & takes a step in that direction. This helper fct is defined here bc FM needs to respond to the users input (which is processed in this fct).
* bool doSquirt() = if FM has water units left, it plays the sound squirt and decrements the # of water units. It then sees if the Squirt object can travel 4 units without hitting a boulder or dirt (if it does it returns false); otherwise, it creates a Squirt object via StudentWorld and returns true
* bool useSonar() = if FM has sonar to use, it decrements # of sonar kits then calls StudentWorld’s makeObjectVisible(int x, int y, int distance) which makes visible all objects a distance of 12 units from x and y
* void getGoodie(GoodieType goodie) = based on the type of Goodie that is passed into the fct, FM increases its score, updates its private members which hold how many goodies overall it has, or calls StudentWorld to update # of barrels gotten
* bool bribeGold() = if FM has a gold nugget to drop, it creates a TempGoldNugget object and returns true, otherwise false
* virtual void hitByBoulder() = derived from Actor and just sets FM to dead using setDead()
* bool handleEscape() = sets FM to dead and returns true
* virtual void annoy() = is called when Protestor shouts at FM; updates hitpoints, plays annoyed FM sound, and determines if FM is now dead; uses Actor’s already declared fct annoy() instead of redefining a fct

the next 4 fcts are used by StudentWorld to update the stats string

* int getGoldNuggets() const = returns # of gold nuggets
* int getNumSquirts() const = returns # of water units
* int getHealth() const = returns health of FM based on hitpoints
* int getSonar() const = returns # of sonar kits
* Boulder (derived from Actor)
* virtual void doSomething() =
  + if the boulder is in stable state & if it canMove() down, set to waiting state & sets sleep ticks to 30, otherwise returns
  + if boulder is in waiting state, sets state to falling & plays falling sound
  + if boulder is in falling state, finds all protestors & FM under boulder & calls hitByBoulder() for each of them if w/in specific distance; also continues to move down until it reaches dirt (then set to dead)
* virtual bool willBlock(int x, int y, bool canDig) = if the distance from the boulder is <= 3, returns true bc it will block any Actor, otherwise false
* Squirt (derived from Actor)
* virtual void doSomething() = annoys protestor object, if it canMove() in current direction it does (if not, is set to dead), else decreases travel distance until it reaches 0 (then is dead)
* Goodie (derived from Actor) = encompass Barrel, Water Pool, Nugget, Oil; receives goodie type in constructor as an enum
* virtual void doSomething() = if distance from goodie to FM is <= 4 & the goodie is not visible, it sets the goodie to visible; otherwise, if distance is <= 3, depending on the goodie a different sound is played & FM calls its getGoodie fct which takes the goodie type enum
* Barrel (derived from Goodie)
* GoldNugget (derived from Goodie)
* TempGoodie (derived from Goodie)
* virtual void doSomething() = calls Goodie’s doSomething() and decLifetime()
* void decLifetime() = decrements the lifetime of the object until it reaches 0, then the object is set to dead; this fct is declared in this class bc it deals w temporary goodies, which have the commonality of a decreasing lifetime and eventual death
* SonarKit (derived from TempGoodie)
* WaterPool (derived from TempGoodie)
* TempGoldNugget (derived from TempGoodie)
* virtual void doSomething() = if can find protestor close to gold nugget, plays sound of protestor finding gold & sets nugget to dead, otherwise calls decLifetime()
* Protestor (derived from Actor)
* virtual void doSomething() = handles all the scenarios a Protestor may encounter: giving up and exiting the maze, shouting at FM (which calls isFacingFrackMan()), following FM if FM is in line of sight, changing directions if numSquaresToMoveInCurrentDirection is <= 0, turning at an intersection, setting numSquaresToMoveInCurrentDirection to 0 if can’t move in current direction any more, and simply continuing in same direction
* virtual void hitByBoulder() = set Protestor to dead, plays give up sound, increases FM score
* virtual bool canbeBribed () const = returns true until protestor gives up
* virtual bool bribeWithGold() = increase FM score, set Protestor to dead
* virtual bool quickSeekFrackMan() = returns false here but is modified in Hardcore
* bool isFacingFrackMan() = determines angle btwn FM and Protestor to determine if Protestor is facing FM; this is a helper fct which takes care of this aspect of Protestor’s doSomething()
* virtual bool canbeAnnoyed() const = returns true until protestor gives up
* bool isFrackManInLineOfSight(Direction &dir, int &newX, int &newY) = checks all directions (up, down, left, right) by calling canMove() until canMove() returns false and determines if FM is in one of those x, y positions along that path; if so, returns true, else false. This function helps with an aspect of the implementation of doSomething() for Protestor.
* bool turnAtIntersection(Direction &dir) = checks turnticks and if <= 0, chooses new direction based on current direction, checks to see if the 2 new directions are navigable; if both are, chooses one randomly, else chooses the one that is clear. This function helps with an aspect of the implementation of doSomething() for Protestor.
* virtual void annoy() = is called when FM squirts a Protestor; decreases # hitpoints, checks to see if Protestor is dead; if it is, plays give up sound and updates FM score depending on if a Regular or Hardcore Protestor is squirted, else plays annoyed protestor sound and sets new sleepticks
* Hardcore (derived from Protestor)
* virtual void bribeWithGold() = increases FM score and sets new sleepticks (does not give up, unlike regular protestors which is why it is virtual)
* virtual bool quickSeekFrackMan() = computes M steps and calls moveInMaze() in StudentWorld which returns int # of steps it takes to reach FM; if steps <= M, it sets Hardcore Protestor’s direction to new direction and takes step in that direction returning true, otherwise returns false
* StudentWorld
* bool clearDirt(int x, int y) = returns true if clears any dirt, but is designed with a nested for loop to clear a 4x4 starting at x, y; if no dirt is cleared, it returns false
* virtual int init() = creates new frackman, initializes intervals for goodie and protestor creation, calls createDirt(), determines # of barrels, creates boulders, gold nuggets, and oil barrels via createInitObjects
* FrackMan\* getPlayer() const = returns pointer to frackman so FM can be access in Actor classes
* void gotBarrel() = is called when FM picks up a barrel 🡪 decreases # of barrels (private member variable)
* static StudentWorld\* getWorld() = returns pointer to \*this for StudentWorld in order to access its public functions
* virtual int move() = checks if FM has died (if so set status player died), checks if all barrels have been found (if so, go to next level), sets the stat text via formatString(), calls FM’s doSomething(), if goodie interval is 0 🡪 creates respective goodie using rand # probability to determine which is created (else decrement the interval), if protestor interval is 0 & # of protestors is < certain value 🡪 creates either regular or hardcore protestor based on rand # probability (else decrement interval), iterates through actors to processTicks(), calls cleanUpAfterMove() which deletes all actors that are dead & clears them from the vector and then continues game
* int moveinMaze(int startX, int startY, int endX, int endY, int &nextX, int &nextY, Direction &newD) = finds solution using queue algorithm from end point to start point; determines # of steps it takes and returns that #; right before returning, it changes nextX and nextY to the position right before the ending point (so the actor knows where to move next); based on the new x and y coord, also determines which direction to move which becomes newD. This function is used to help Protestors navigate toward the exit and for Hardcore Protestors to find FM.
* void makeObjectsVisible(int x, int y, double radius) = sets all objects to visible if they already aren’t if they are within the specified radius from x, y. This function allows sonar to work.
* Actor\* findClosestProtestor(int x, int y) = returns actor pointer (which is really a protestor) within 3 units of x, y
* bool canMove(int &x, int &y, bool canDig, Direction dir, Actor\* ignore = nullptr) = very useful function that returns true if a certain object can move in the given direction if they start at x, y; takes parameter canDig to allow for special FM case so it can return true even if there is dirt there; takes parameter Actor\* ignore to prevent boulder from blocking itself when it falls
* virtual void cleanup() = deletes all actor objects & clears actor vector deletes frackman, deletes dirt array
* void createSquirt(int x, int y, Direction dir) = creates a squirt object at x, y in dir direction and pushes it back into actor vector; is called when FM creates a squirt
* void createTempGoldNugget(int x, int y) = creates a tempgoldnugget object at x, y in dir direction and pushes it back into actor vector; is called when FM drops a gold nugget

1. LIST OF FAILED FINISHED FUNCTIONALITY
2. DESIGN DECISIONS/ASSUMPTIONS MADE

* I saw commonality between Regular Protestors and Hardcore Protestors and among the different Goodies right away so I decided to make overarching classes for those immediately. However, it was ambiguous to decipher Protestors/FM from the other Actor objects (which I needed to do in order to perform certain actions/respond differently), so I created virtual bool functions in the Actor class that told me which Actor object it was. This was tricky for me to realize immediately, since I knew you could not directly pass pointers. Similarly with the willBlock() and canMove() functions, I was able to allow FM to dig dirt only because he can dig but have dirt block every other object. And in the Boulder class, I changed the implementation of willBlock() to block every object.
* Within Protestor’s doSomething(), if the protestor is hit by a boulder, the protestor can be trapped trying to change directions since the protestor doesn’t move every tick. This case is checked for by seeing if the Protestor can move in every direction. If it can’t move in any direction, simply return and do not do anything that tick until it dies. Otherwise, pick a direction and continue.

1. TESTING FOR EACH CLASS

* Actor = base class
* Dirt = It doesn’t really do anything, but when I created m\_dirt array of Dirt pointers in StudentWorld, I made sure the Dirt showed up properly and that the spaces where there should not be Dirt were nullptr
* FrackMan
  + I began by first making sure FrackMan started at the right place in the maze when I started the game. Then I implemented taking in user input, which required ploughDirt() which called canMove() in StudentWorld. I then played the game to make sure that the bounds were being checked in the dirt array and that the correct sound was being made, etc.
  + Then I implemented the rest of doSomething() (the helper fcts doSquirt(), bribeGold(), etc.). I tested out the “z” key by going to a part of the field and pressing to see if the objects around it showed up. I also tested the escape key by simply pressing it and making sure there were no glitches when trying to return to play again. I tested each part of doSomething() once I had the respective classes finished and functioning to ensure that the way the class and FM interacted was correct (squirt, grabbing goodies, etc.)
* Boulder
  + At first I tested to make sure that the boulders first showed up in the dirt, with the dirt properly cleared around it. This required proper implementation of getting random coordinates and isFilled(). I also took FM and made sure that the boulder blocked my movements.
  + I tested the falling boulders by letting FM clear the dirt underneath them and making sure that they fall, make a sound, and disappear once they reach dirt or a boulder.
* Squirt
  + I tested squirt by pressing the spacebar directly at Protestors (ensuring the points were decreasing properly and that all sounds were made) at the start of the game. I also tested it against dirt, boulders, and nearby protestors to make certain that the object itself is not created (only the sound is made and the # of water units is decremented)
* Barrel & GoldNugget (normal Goodie)
  + I tested both of these at the same time. When I first embedded them into the dirt, I made them visible to make sure that they actually showed up and were at least 6 units away from each other (including the boulders).
  + Later with the FM, I made sure that they appeared when FM got within a specific radius of them and that the sound was made and points added when FM picks up these goodies. I stepped through the debugger at this time too to see if these objects were deleted in StudentWorld.
* WaterPool & SonarKit (temp Goodie)
  + I also tested these both at the same time since they have the same implementations. I just started playing the game and waited for sonar kit and water pools to show up. If they weren’t showing up like they were supposed to, I looked back at my move() fct to check to see if the ticks were decrementing properly. None of these objects were appearing in the middle of the dirt which meant that the canMove() fct was doing its job properly as well.
  + Once they were showing up properly, I went to grab them and made sure (just like the barrel and gold nugget goodies) that the sound played and that the proper stats were incremented.
* TempGoldNugget
  + FM drops a gold nugget and it appears properly at that spot. I wait for a Regular Protestor to come and see if it registers (the bribe sound is played and the scores are increased) and leaves properly. I do the same with the Hardcore Protestors but this time ensure that the protestor does not leave because it cannot be bribed.
* Protestor
  + I started playing and followed the movements of the protestors. At one point they were getting stuck, so I stepped through with the debugger and found that I had accidently made a parameter pass by value instead of reference, which caused all sorts of problems. The Protestors were also not leaving properly, so I stepped through and saw that marking of the path with “X” was wrong and needed to be changed.
  + I also tested to see if they die when a boulder is dropped on them, when I present them with gold, and when I squirt them 3 times.
* Hardcore
  + Hardcore protestors are similarly tested to regular protestors; however, I had to step through my quickSeekFrackMan() function which had moveInMaze() to be sure that the right # of steps and the right path is taken toward FM, otherwise the protestor cannot do its job.
  + I also presented it with gold and made sure that it was not bribed (did not give up and leave the field like normal protestors) and squirted it to see if it gives up and leaves appropriately.