Cynthia Nguyen

705767700

CS32

TunnelMan Report

Hierarchy description: Everything is a derived directly from Actor, except HardProtest, which is derived from RegProtest, which is derived from Actor.

**Descriptions of class functions**

* StudentWorld:
  + virtual int init();
    1. set random seed for round
    2. initialize timers and variables for the level
    3. fill in oil field with earth objects except for top layer and tunnel
    4. randomly generate objects(Oil, Gold, Boulder)
       - generate random x, y coordinates for object
         * check if there is already an object in 6.0 radius
         * pushback object to vector of Actor pointers
    5. return continue game
  + virtual int move();
    1. update game text
    2. Add new actors for tick
       - Randomly checks if a goodie can spawn this tick (1/G chance)
         * If can spawn, check if it is sonar (1/5 chance)

If sonar, check if there is any other items in 6.0 radius

If sonar cant spawn, spawn a water

* + - * + Spawn water

Check if it is in a clear air pocket and not within 6.0 radius of an item

* + - * Spawn Protestors
        + If it has reached time to spawn a new protestor

Check if number of protestor is less than max number of protestor for level

Reset time since last spawned

Randomize according to probability of hardcore protestor

Add protestor according to randomizer

* + 1. Player does something, then check if player is dead
    2. Iterate through vector of Actor pointers
       - Have them do something
       - Check if player is dead
    3. Remove all dead actors
       - Erase from vector
    4. Check if player is dead
    5. If number of barrels is 0, finish the level
    6. Otherwise continue game
  + virtual void cleanUp();
    1. delete Player
    2. delete all remaining earth objects
    3. delete all remaining objects in vector of Actor pointers
  + void dig();
    1. get Player’s x and y coordinates
    2. create flag to see if Player dug anything
    3. using double for loop delete any remaining Earth object Player walks into
    4. if flag marked that the player dug, play dig sound
  + void clearSquare(int x, int y);
    1. deletes earth objects in the square with x and y as the bottom leftmost coordinate using double for loop
  + Earth\* getField(int x, int y);
    1. Gets the value of the array of Earth object at coordinates x and y
  + TunnelMan\*& getPlayer();
    1. Gets the pointer to the player
  + std::vector<Actor\*>& getObj();
    1. Gets the vector of Actor Objects
  + void grabOil();
    1. decreases the number of oil on the fiels
  + const bool isclear(int x, int y);
    1. uses double for loop to check that the 4x4 square with x and y at the left bottom most corner has no earth objects and no boulders
  + bool isBoulder(int x, int y);
    1. checks that there is no stable boulder in the 4.0 radius of x and y
  + void deadProtestor();
    1. decrements the number of protestors on the field
* Actor:
  + StudentWorld\* getWorld();
    1. returns current World
  + const bool isdead();
    1. returns if is dead
  + void setdead();
    1. sets actor as dead
  + double radius\_dist(Actor\* one, Actor\* two);
    1. finds the radius between the center of two actors
  + double radius\_dist(int x1, int y1, Actor\* two);
    1. find the radius between the center of the 4x4 square at x1, y1, and an actor
  + const int getID();
    1. returns actor ID
  + const int getHealth();
    1. returns actor's health
  + const bool isannoyed();
    1. returns if is annoyed
  + virtual void annoy(int points);
    1. decrease hit points by parameter points
    2. is virtual because when character becomes completely annoyed, there will be different sounds/actions
  + virtual bool inProtestorRadius(int x, int y, double radius, int index);
    1. returns true if there is a protestor withing the given radius of x and y, and gives the index of the Protestor in that radius
       - is virtual because there are different actions to do when a protestor is in the radius
* Earth:
  + N/A
* TunnelMan:
  + Member variable
    1. int water
       - Squirts for tunnel man
    2. int sonarCharge
       - number of usable sonar charges in inventory
    3. int gold
       - number of gold in inventory
  + void doSomething();
    1. check that player isn’t dead
    2. switch to get keys
       - direction keys
         * if isn’t facing same direction as in key, set direction, break
         * if is at a border, return
         * adjust x or y coordinate according to direction
       - space key
         * if there is water in inventory

get direction of player

if four blocks in player direction is clear

add Squirt object

decrease number water in inventory

play squirt sound

* + - * escape key
        + set hitpoints to 0
        + set player to dead
      * ‘Z’ or ‘z’
        + if there is sonar in inventory

decrease sonar charge in inventory

iterate through vector of Actor pointers

set every actor in 12.0 radius to visible

* + - * tab key
        + if there is gold in inventory

place a piece of gold at player’s coordinates with parameter dropped as true (prevents player from picking it back up)

decrease number of gold in inventory

* + 1. if is not blocked by boulder at coordinate for next move
       - move player to next coordinates
       - dig the dirt in the player’s area
  + void grabGold();
    1. add gold to inventory
  + void grabSonar();
    1. add sonar to inventory
  + void grabWater();
    1. add water to inventory
  + void annoy(int points);
    1. decrease health by parameter points
    2. if fully annoyed
       - play correct sound
       - set player dead
  + gets number in inventory
    1. const int getWater();
    2. const int getSonar();
    3. const int getGold();
  + bool TunnelMan::blocked(int x, int y)
    1. checks if there is a boulder within a 4.0 radius of x and y
* Boulder:
  + Member variables
    1. int wait\_time
       - holds number of ticks waiting since earth has been cleared underneath
  + void doSomething();
    1. check that boulder isn’t dead
    2. if wait\_time is greater than or equal to 30
       - boulder falls
    3. else if boulder is not stable, increase the wait time
    4. do nothing if boulder is stable
  + bool stable(int x, int y);
    1. checks if there is no earth right underneath the boulder
  + void inProtestorRadius(int x, int y, double radius);
    1. if there is a protestor within the 3.0 radius of boulder’s coordinates
    2. annoy protestor by 100 points
    3. if protestor is annoyed
       - increase score by 500 points
  + void fall();
    1. set dead if hits earth or bottom of field
    2. otherwise move down a unit
       - if this move hits a player
         * annoy player by 100 points (completely annoy and kill player)
       - use inProtestorRadius to properly annoy any protestors the boulder hits in move
* Squirt:
  + Member Variables
    1. int distance
       - distance left to travel
       - initialized at 4
  + void doSomething();
    1. if inProtestorRadius //function also properly annoys any protestors
       - set squirt as dead
    2. if distance is 0 or squirt hits an earth or boulder or edge
       - set dead
    3. otherwise move next coordinate according to direction
  + bool inProtestorRadius(int x, int y, double radius);
    1. returns true if squirt hits at least one protestor
    2. finds all protestor within 3.0 radius of squirt
    3. annoy protestors by 2 points
       - if regular protestor is annoyed, increase score by 100
       - if hardcore protestor is annoyed, increase score by 250
* Barrel
  + void doSomething();
    1. check that is isn’t dead
    2. if it isn’t visible and within 4.0 radius of player, set visible
    3. if is visible and within 3.0 radius of player
       - set dead
       - play SOUND\_FOUND\_OIL
       - increase score by 1000
       - tell studentworld to decrease num of oil on field
* Gold:
  + Member variables
    1. bool dropped
       - tells if gold was dropped by player or not
       - initialize by parameter
    2. int drop\_time
       - counts number of ticks since player dropped gold
  + void doSomething();
    1. check gold isn’t dead
    2. if gold has been dropped 100 ticks
       - set dead
    3. else if gold isn’t visible and within 4.0 radius of player
       - set visible
    4. else if gold is visible and within a 3.0 radius of player and is not dropped by player
       - set gold as dead
       - play SOUND\_GOT\_GOODIE
       - increase score by 10
       - increase player’s inventory by one gold
    5. else if gold is dropped by player and is in a protestor’s radius of 3.0
       - set gold dead
       - play SOUND\_PROTESTOR\_FOUND\_GOLD
       - bribe the protestor accordingly (regular or hardcore)
       - increase score accordingly (regular = 25 or hardcore = 50)
    6. else if gold is dropped
       - increase drop\_time
* SonarKit:
  + Member Variables
    1. int time
       - ticks since spawned
    2. int maxTime
       - max ticks SonarKit is available for
  + void doSomething();
    1. check if is dead
    2. set dead if reached max ticks
    3. if within a 3.0 radius of player
       - set dead
       - play SOUND\_GOT\_GOODIE
       - increase player’s inventory with a sonar
       - increase score by 75
    4. else increase number of ticks since spawned
* WaterPool:
  + Member Variables
    1. int time
       - ticks since spawned
    2. int maxTime
       - max ticks WaterPool is available for
  + void doSomething();
    1. check if is dead
    2. set dead if reached max ticks
    3. if within a 3.0 radius of player
       - set dead
       - play SOUND\_GOT\_GOODIE
       - increase player’s inventory with a water
       - increase score by 100
    4. else increase number of ticks since spawned
* RegProtest:
  + Member Variables
    1. int restTime
       - number of ticks rested
    2. int sinceShout
       - number of ticks since last shout
    3. int sincePerp
       - number of ticks since last perpendicular turn
    4. int numMove
       - random number between 8 and 60 to move in particular direction
    5. vector<pair<int, int>>path;
       - holds a path for protestor to follow
    6. int maze [61][61]
       - maze of steps to designated exit point
  + virtual void doSomething();
    1. if have not reached movement tick
       - increase restTime
    2. otherwise
       - reset TimetoRest
       - reset restTime
       - increase sinceShout
       - increase sincePerp
       - check if is dead
       - if annoyed
         * follow path to 60,60
         * if reached 60,60

set dead

decrease number of protestor on field

* + - * if within 4.0 radius of player and facing player direction
        + if sinceShout > 15

shout at player

annoy player by 2 points

reset sinceShout

return

* + - * if is player is in horizontal or vertical vision of protestor (even if not facing) and radius between protestor and player is greater than 4.0
        + face the player, move towards the player
        + reset numMove
        + return
      * else decrease numMove
        + if numMove <= 0
        + find new possible direction
        + randomize numMove between 8 and 60
      * if sincePerp> 200 and is able to move perpendicular
        + reset sincePerp
        + set perpendicular direction
        + randomize numMove between 8 and 60
      * if next move is clear of earth and boulders and not outside of border
        + move to next coordinates
      * else reset numMoves for new direction
  + void annoy(int points);
    1. if not annoyed
       - decrease hitPoints by points
       - if health <= 0
         * set annoyed
         * play SOUND\_PROTESTOR\_GIVE\_UP
         * reset restTime
         * create a maze with exit (60,60) for optimal path

also creates path

* + - * + reset TimetoRest
      * else
        + play SOUND\_PROTESTOR\_ANNOYED
        + change TimetoRest accordingly
  + virtual void bribe();
    1. annoy protestor so that he leaves the maze
    2. chose to make this virtual because hardcore protestors behave differently when bribed
  + bool ishorizontal(Direction& next\_dir, int& next\_x);
    1. checks if player and protestor are on the same Y axis
    2. checks if there is any earth or boulder between player and protestor
    3. if its clear
       - set next\_dir to face player and next\_x to move towards player
    4. params don’t change if returns false
  + bool isvertical(Direction& next\_dir, int& next\_y);
    1. checks if player and protestor are on same X axis
    2. checks if there is earth or boulder between player or protestor
    3. if its clear
       - set next\_dir to face player and next\_y to move towards protestor
    4. params don’t change if returns false
  + bool isperp(Direction& next\_dir);
    1. if currently going up or down
       - if both left and right are clear, randomly pick left or right to for next\_dir
       - if only one is clear, next\_dir is the clear direction
    2. if currently going left or right
       - if both up and down are clear, randomly pick up or down for next\_dir
       - if only one is clear, next\_dir is the clear direction
    3. returns false if there is no clear direction
  + void followPath();
    1. if path is not empty
       - get coordinates from back of path vector
       - pop back vector
       - set direction according to new coordinates
       - move to new coordinates
  + bool faceingPlayerDir();
    1. get direction
       - if direction is facing player
       - return true
    2. return false otherwise
  + void findNewMove(Direction& next\_dir);
    1. infinite loop for a random direction
    2. if one move in the direction is clear
       - set direction to that move
       - randomize numMove between 8 and 60
       - break loop
  + bool createMaze(int exit\_x, int exit\_y);
    1. use BFS to fill in a maze with ints
       - start point has int 0
       - fill in all empty adjacent cells with (maze[x][y] + 1)
    2. once reach exit
       - call findPath to decode maze
       - return true
  + void findPath(int exit\_x, int exit\_y);
    1. starting with the exit, step == value at maze[x][y]
    2. while steps >=0
       - grab last coordinates in path
       - step is int at maze[x][y]
       - if step == 1
         * return, this will be protestor’s next step
       - find adjacent coordinates where int steps is one less than the steps at maze[x][y]
       - push coordinates into vector
  + void resetMaze();
    1. fill in maze with ints -1
* HardProtest:
  + Member variables
    1. int M
       - maximum number of steps to path to Player
  + void doSomething();
    1. if have not reached movement tick
       - increase restTime
    2. otherwise
       - reset TimetoRest
       - reset restTime
       - increase sinceShout
       - increase sincePerp
       - check if is dead
       - if annoyed
         * follow path to 60,60
         * if reached 60,60

set dead

decrease number of protestor on field

* + - * if within 4.0 radius of player and facing player direction
        + if sinceShout > 15

shout at player

annoy player by 2 points

reset sinceShout

return

* + - * if greater than 4.0 radius of player
        + reset the maze
        + create maze with exit as Player’s coordinates
        + number of steps to player is <= M

follow the path to player

return

* + - * if is player is in horizontal or vertical vision of protestor (even if not facing) and radius between protestor and player is greater than 4.0
        + face the player, move towards the player
        + reset numMove
        + return
      * else decrease numMove
        + if numMove <= 0
        + find new possible direction
        + randomize numMove between 8 and 60
      * if sincePerp> 200 and is able to move perpendicular
        + reset sincePerp
        + set perpendicular direction
        + randomize numMove between 8 and 60
      * if next move is clear of earth and boulders and not outside of border
        + move to next coordinates
      * else reset numMoves for new direction
  + void annoy(int points);
    1. if not annoyed
       - reduce hitpoints by parameter points
       - if hitpoints <= 0
         * set annoyed
         * play SOUND\_PROTESTOR\_GIVE\_UP
         * reset restTime
         * reset the maze
         * create maze with (60,60) as exit
         * reset TimetoRest
       - else
         * play SOUND\_PROTESTOR\_ANNOYED
         * set TimetoRest for ticks stunned
  + void bribe();
    1. set TimetoRest for ticks to stare

**Unfinished Sections**

I finished all functionalities.

**Testing**

TunnelMan:

I tested TunnelMan by digging every earth block, and walking to every border. I also tested each key, squirting, dropping gold, and using my sonar kit. I also tried to walk back over the gold that I dropped to see if I could pick it up. I walked directly towards boulders, and checked every angle of approach to the boulder to ensure there is no overlap with the boulders. I tested walking up to protestors to get shouted at until I died. I also tested dropping a boulder onto myself to see if I died. I also played 11 levels of the game to see if the behavior remained correct.

Earth:

I tested Earth by selectively deleting coordinates on my Field to ensure that every coordinate correctly corresponds to each Earth element. I also dug every Earth object to make sure that there would be no memory leaks or overflow. I also squirted water at Earth objects to test that it would properly kill the squirt object. I had boulders fall only different amounts of Earth objects to make sure that the it would properly kill the boulders. I also called the Earth’s doSomething() to make sure there would be no changes to the program.

Boulder:

I tested the boulder by walking into the boulder at several directions and angles to ensure I can not overlap my TunnelMan with the boulder. I also dropped the Boulder on several protestors. Initially the protestors could not navigate back to the exit because the falling boulder served as an obstacle, but I changed my code so that the falling boulder is not considered and obstacle, and the protestors was able to properly exit the maze. I also tested that the boulder would break at any touch of Earth or reach the bottom. I tested the boulder could hurt multiple protestors while falling by dropping it on two protestors. I dropped the boulder on both hardcore and regular protestors. I also squirted water at boulders to ensure that the boulders would kill squirt objects. For boulder spawning, I tested 40-45 boulders being spawned at a single level rather than the given formula to see if there would be any boulder overlap, which is around the max number of boulders that can fit in an oil field.

Squirt:

I tested Squirt by giving myself 100 waters. Then I proceeded to Squirt at clear paths, then at earth objects, and then at boulders, and at pockets that are either semi filled with earth or with boulders. Then I squirted protestors, counting the number of squirts to annoy a protestor is accurate. I also squirt water at different levels in the oil field to ensure that there is no issue at different coordinates. I also squirt at the borders to make sure that the object can not leave the field.

Barrel:

I tested Barrel by first testing spawning. I temporarily set all Barrels to spawn as visible, and spawned 50 Barrels to ensure there is no overlap. Then I reverted to the original state of the Barrel for game testing. Then I tested by digging slowly to see how far away I am until the Barrel becomes visible, and cross check it with the sample. I check that if I dig one block closer from where it became visible, I would collect the Barrel and the stat would decrease by one. I also allowed protestors to walk towards and past Barrels to make sure that it would not be an obstruction. I played multiple games to test that once every Barrel is found, the game ends.

Gold:

I tested Gold by spawning 50 golds while it was temporarily set as visible during spawn to check that there is no overlap. Then I reverted to the original state of Gold for game testing. I tested by digging slowing to see how far away I am until the Gold becomes visible, and I cross check with the sample game. Then I check if digging one block closer would mean collecting the gold. I also tested leaving the gold out and undropped and luring a protestor to walk past to make sure that Protestors can only picked up dropped Gold. Then I test dropping the Gold and walking through it to make sure the player can not collect dropped gold. I also checked to make sure the stats text would properly update while dropping and collecting gold. Then I test dropping the Gold to make sure the protestors pick it up. I checked that the protestor behaved properly while bribed, and the score increased by the proper amount. I tested dropping the Gold far away from the protestors so that it would disappear after the set amount of ticks.

SonarKit:

I tested SonarKits by changing the times it would spawn to every tick. Then I would leave it running, and after a couple minutes, I would try to collect the SonarKits. Initially, I would get multiple kits after a couple of minutes, but then I adjusted my code so that SonarKits can not spawn in the case that there are already a goodie within a 6.0 radius. Then, I would only collect one Sonar Kit at a time. I also tested that only I can pick up SonarKits by luring protestors through the SonarKit. I also checked that every SonarKit I collected was updated into my inventory and the stats text.

WaterPool:

I tested WaterPoop by changing the times it would spawn to every tick. Then I would see if there was any WaterPools that overlapped with any other Goodies or Earth or Boulder objects. Then I tested walking withing 3.0 radius of the WaterPool to collect it. I also had Protestors walk through WaterPools to ensure that they would not collect any WaterPools. I also checked that with every WaterPool I collect, I gain 5 pieces of Water to my inventory.

RegProtest:

I tested Regular Protestors by squirting at them repeatedly with water to annoy them. I would do this to test to see if they would take the optimal path. I put them into loops and annoyed them several times and watched to make sure it took the most optimal path. I also tested in random structures as well. I counted to make sure that it was consistent amount of squirts to annoy a protestor. I also dropped boulders on Regular Protestors to make sure they get annoyed and exit the field. I also tested the horizonal and vertical vision by creating tunnels and peeping in to see if the Regular Protestor would change direction. I tested the perpendicular movement by allowing the game state text to say “test” when the function is called. I tested it’s movements by hiding the TunnelMan and watching it move around, making sure it would not walk into boulders, earth, or out of bounds.

HardProtest:

I tested Hardcore Protestors by squirting at them repeatedly with water to annoy them. I would test to see if it would take the optimal path by putting them in loops and annoying them within the loops. I would then see if it would take the most optimal path out of the loop. I also tested in random structures as well. I counted to make sure that it was a consistent amount of squirts to annoy a Hardcore protestor, but still more squirts than a Regular Protestor. I would also drop boulders to make sure that the Hardcore protestor would get annoyed and leave the field. I tested the horizontal and vertical functions by making tunnels and peeping through to see if the Hardcore Protestor would change direction and walk towards me. I also tested the tracking by creating a loop, and inching closer to the Hardcore protestor to see if it would change directions to chase me. I tested the perpendicular movement by allowing the game state text to say “test” when the function is called. I tested its regular movements by hiding the TunnelMan, and watching the protestor move around, making sure it does not walk int boulders, earth, or out of bounds.

Ambiguity:

* I had the protestor started so that it would be able to shout on the first tick.
* I had the protestor started so that it would be able to move perpendicular on the first tick.
* Because there can not be multiple SonarKits at a time, I had it so if sonar cant spawn, a WaterPool will spawn instead
* The samples showed different ways the TunnelMan can interact with Boulders, so I followed the Mac way, where it can get near the boulder and within the 4x4 square, but it can not overlap with the boulder image. This is with the radius between the center of the Player and the Boulder as greater than 4.0.