Professor Smallberg – Lecture 2 Leo Gretzinger

Project 3 Report 2/28/19

**High Level Description of Classes:**

Actor class:

* doSomeThing(). This class allows each actor to do their specified action at every move of the game. I defined it as pure virtual because every actor uses doSomeThing() but in a different way
* isDead() checks if the actor in question is dead. Implemented the same way for every actor.
* setDead() sets the actor to dead. Implemented the same way for every actor
* getWorld() returns the Student World that the actor exists in. Implemented the same way for every actor and is used to connect the two trees
* activateIfAppropriate() – if the actor using this function is activated, then perform its effect on a specified actor
* useExitIfAppropriate() – if the actor uses exits, use the exit
* dieByFallOrBurnIfAppropriate() – if the actor can die by falling into a pit or burning, this function causes it to die
* beVomitedOnIfAppropriate() – if the actor can be vomited on by a zombie, this function causes it to receive the vomit (generally the actor gets infected and his infection count starts)
* pickUpGoodieIfAppropriate() – if the actor can pick up goodies (Penelope only) then it should pick the specified goodie
* isBlockingType() returns whether the actor is a blocking type or not (wall or agent). This is implemented as returning false for every actor besides the ones that have it implemented as returning true
* blocksFlame() returns whether the actor blocks flame or not. This is implemented as returning false for every actor besides the ones that have it implemented as returning true
* isntFlammable() returns whether the actor is flammable or not. This is implemented as returning false for every actor besides the ones that have it implemented as returning true
* canPickUp() returns whether the actor can pick up goodies or not. This is implemented as returning false for every actor besides the ones that have it implemented as returning true
* triggersOnlyActiveLandmines() returns whether the actor triggers active landmines or not. This is implemented as returning false for every actor besides the ones that have it implemented as returning true
* triggersZombieVomit() returns whether the actor triggers zombie (humans) or not. This is implemented as returning false for every actor besides the ones that have it implemented as returning true
* threatensCitizens() returns whether the actor threatens citizens (zombies) or not. This is implemented as returning false for every actor besides the ones that have it implemented as returning true
* triggersCitizens() returns whether the actor triggersCitizens (all agents except citizens) or not. This is implemented as returning false for every actor besides the ones that have it implemented as returning true

Wall Class:

* doSomeThing() – wall’s doSomeThing doesn’t do anything, it just returns.
* isBlockingType() – wall is a blocking type
* blocksFlame() – wall blocks flame
* isntFlammable() – wall is not flammable

Exit Class:

* doSomeThing() – if exit is not dead, calls the activateAllAppropriateActors from StudentWorld class
* blocksFlame() – exit blocks flame
* isntFlammable() – exit is not flammable
* activateIfAppropriate() – calls useExitIfAppropriate if the actor uses exits

Pit Class:

* doSomeThing() – If pit is not dead, calls the activateAllAppropriateActors from StudentWorld class
* activateIfAppropriate() – Calls dieByBurnOrFallIfAppropriate if the actor dies by pit. Pits cause people to fall and die

Projectile Class:

* isntFlammable() – All projectiles are not flammable
* increaseTicks() – Increase the ticks(how many frames the projectile has been alive) by one
* descreaseTicks() – Decrease the ticks by one
* getTicks() – Returns the ticks of the projectile

Flame Class:

* threatensCitizens() – Flame threatens citizen
* doSomeThing() – If flame is not dead (if been alive for more than 2 ticks die) calls the activateAllAppropriateActors from StudentWorld class
* activateIfAppropriate() – Calls dieByBurnOrFallIfAppropriate if the actor dies by flame. Fames cause people to burn and die

Vomit Class:

* doSomeThing() – If Vomit is not dead (dies if been alive for more than two ticks) calls the activateAllAppropriateActors from StudentWorld class
* threatensCitizens() – Vomit threatens citizens
* activateIfAppropriate() – Calls setInfected if the actor can be vomited on. Vomit infects people

Landmine Class:

* doSomeThing() – If Landmine is not dead (dies if been alive for more than thirty ticks) calls the activateAllAppropriateActors from StudentWorld class
* activateIfAppropriate() – Calls dieByBurnOrFallIfAppropriate if the actor dies by flame and on itself. Landmines create flames that cause people to burn and die
* dieByFallOrBurnIfAppropriate() – Kills the landmine and creates a flame on the location and in the eight places all around it. Also creates a pit on location
* setTicks() – Sets ticks to a specified value
* setInactive() – Sets the landmine to inactive

Goodie Class:

* activateIfAppropriate() – Calls pickUpGoodie if the actor picks goodies up. Goodie doesn’t implement doSomething because the individual goodies doSomeThing
* dieByFallOrBurnIfAppropriate() – Kills the goodie
* pickUp() – Is pure virtual because each goodie has a different effect when picked up

Vaccine Goodie:

* doSomeThing() – calls activateOnAppropriate from Student class
* pickUp - increases the score by 50 and increases the amount of vaccines by 1

Gas Can Goodie:

* doSomeThing() – calls activateOnAppropriate from Student class
* pickUp - increases the score by 50 and increases the amount of flamethrower by 5

Landmine Goodie:

* doSomeThing() – calls activateOnAppropriate from Student class
* pickUp - increases the score by 50 and increases the amount of landmines by 2

Agent Class:

* isBlockingType() – agent blocks
* triggersOnlyActiveLandmines() – Agent triggers landmines
* computeNewCoords() – compute new coordinates in a specified direction and distance

Human Class:

* triggersZombies() – Humas trigger zombies
* beVomitedOnIfAppropriate() – Human gets infected when they are vomited on
* clearInfection() – Sets the infection count to 0
* getInfected() – returns whether the human is infectedor not
* increaseInfection() – Increases the infection by one
* getInfectionDuration() – returns the infection duration
* setInfected() – sets the human to infected
* setUninfected() – Uninfects the human

Penelope Class:

* doSomeThing() – largely determined by user input, either go up, down, left or right, or drop a landmine, shoot fire or use a vaccine
* useExitIfAppropriate() – uses the exit if all the citizens are gone
* dieByFallOrBurnIfAppropriate() – die when burned or falls
* pickUpGoodie() – picks up a goodie
* triggersCitizens() – Penelope triggers citizens
* canPickUp() – Penelope picks stuff up
* increaseVaccines() – increases vaccines by one
* decreaseVaccines() – decreases vaccines by one
* increaseFlameCharges() – increases flames by 5
* decreaseFlameCharges() – decreases flames by 1
* increaseLandmines() – increases landmines by 2
* decreaseLandmines() – decreases landmines by 1
* getNumVaccines() – returns number of vaccines
* getNumFlameCharges() – returns the number of flame charges
* getNumLandmines() – returns the number of landmines

Citizen Class:

* doSomeThing() – either moves closer to Penelope or away from a zombie whichever one is closer
* useExitIfAppropriate() – exits the game and decreases number of citizens
* dieByFallOrBurnIfAppropriate() – die from a burn or fall
* incresaeTicks() – increase ticks by 1
* decreaseTicks() – decrease ticks by 1
* getTicks() – returns ticks
* moveToIfWorks() – moves to a location if can

Zombie Class:

* triggersCitizens() – Penelope triggers citizens
* incresaeTicks() – increase ticks by 1
* decreaseTicks() – decrease ticks by 1
* getTicks() – returns ticks
* setMoveDist() – moves a set distance plan
* decreaseMoveDist() – decrease the movement plan
* getMoveDist() – returns the movement plan

Dumb Zombie:

* doSomething() – essentially moves randomly around the screen
* dieByFallOrBurnIfAppropriate() – dies if falls or burns and there’s a 1 in 10 chance if releases a vaccine

Smart Zombie:

* doSomething() – moves according to locations of citizens and Penelope
* dieByFallOrBurnIfAppropriate() – dies if falls or burns
* findNewDir() – finds if citizen or Penelope if around

StudentWorld:

* init() – initializes each new level
* move() – called every tick, basically gives each actor a chance to dosomething each frame
* cleanup() – removes all the actors from the game
* addActor() – adds an actor to the list of actors
* recordCitizenGone() – decreases the number of citizens
* recordLevelFinishedIfAllCitizensGone() – tells the student world that the level is over
* activateOnAppropriateActors() – loops through all actors and activates all that overlap if they’re alive
* isBlocked() – is penelope blocked at this location
* isZombieOrCitizenBlocked() – returns whether the zombie or citizen is blocked
* blockages() – used in determining blocking coordinates
* isFlameBlockedAt() – returns whether the flame is lbocked or not
* isZombieVomitTriggerAt() – returns whether a zombie trigger is at x y
* locateNearestVomitTrigger – locates the nearest zombie trigger (humans) based on eucledian distance
* locateNearestCitizenTrigger() – locates the closest zombie and human and compares by using eucleidian distances
* locateNearestCitizenThreat() – locates the closest zombie using distances, helpful with locatedNearestCitizenTrigger()
* overlapWithAnything() – does this actor overlap with any actor?
* Overlap() – do the two specified actors overlap?
* overlapXY() – does the specified actor and x y overlap

**Missing Functionality:**

* To the best of my knowledge, I think I completed every functionality.

**Designer’s Decisions:**

* I decided to create a list of actors and a separate Penelope object
* The Flame class was unclear about whether it shouldn’t overlap or is blocked by certain objects so I used overlap because there were instances in the example game that used this
* I created a Projectile subclass to encompass Flame, Vomit and Landmine
* I created an Agent subclass which encompassed humans(Penelope and citizens) and zombies(dumb zombie and smart zombie)
* I created an activating object to include all the objects to be activated.

**Test Cases:**

* I tested all the most derived classes:
  + Wall – I tested if it blocked everything at every angle, direction and spacing
  + Exit – I tested if it allowed citizens in at any time, Penelope in only when all the citizens were gone and allowed nothing else and was never overlapped or killed
  + Pit – I tested if it killed all agents
  + Flame – I tested if it killed all agents and goodies and didn’t overlap with walls
  + Vomit – I tested if it infected all humans and did not overlap with anything else
  + Landmines – I tested if they were created correctly, if their timers were set correctly and that they created 9 new flames and a pit object when destroyed (either when the time went off or by flame)
  + Vaccine Goodie – I tested if it increased the score and amount of vaccines correctly
  + Gas can goodie – I tested if it increased the score and amount of flame thrower charges correctly
  + Landmine goodie – I tested if it increased the score and amount of landmines correctly
  + Penelope – I tested if Penelope was blocked by walls and all other agents always, if she could correctly shoot flames, place landmines and use vaccines.
  + Citizen – I tested if the citizen would follow Penelope correctly and run away from all zombies correctly and that it wouldn’t overlap and it would be blocked by walls
  + Dumb Zombie – I tested if the zombie moved randomly every other tick, and didn’t overlap and was blocked correctly and vomited if near enough to a human
  + Smart Zombie – I tested if the zombie moved correctly based on the location of Penelope and the citizens and that it wouldn’t overlap of be blocked incorrectly and that it would spit vomit in the correct situations.