CS 32 Project 3 Report

Junhong Wang - 504941113

**A high-level description of each of your public member functions in each of your classes, and why you chose to define each member function in its host class; also explain why (or why not) you decided to make each function virtual or pure virtual. For example, “I chose to define a pure virtual version of the sneeze() function in my base Actor class because all actors in Zombie Dash are able to sneeze, and each type of actor sneezes in a different way.”**

|  |
| --- |
| **Actor** |
| *Virtual void doSomething() = 0* |
| Actors will doSomething() every tick.  I chose it to be pure virtual function because every actor will be implementing doSomething() |
| *Virtual bool isAlive() = 0* |
| Actors know if they are alive.  I chose it to be pure virtual function because every actor will be implementing isAlive() |
| *Virtual void die() = 0* |
| Actors will set theirs states to be dead.  I chose it to be pure virtual function because every actor will be implementing die(). |
| *Virtual bool isOverlappable()* |
| Return true if the actor can overlap with other actors.  I chose it to be virtual function because only some actors are overlappable. By default, this function returns false. |
| *Virtual bool isFlammable()* |
| Return true if the actor can be killed by Flame.  I chose it to be virtual function because only some actors are flammable. By default, this function returns false. |
| *Virtual bool isPerson()* |
| Return true if the actor is a person. This will be used by Zombies to chase people.  I chose it to be virtual function because only some actors are people. By default, this function returns false. |
| *Virtual bool isZombie()* |
| Return true if the actor is a person. This will be used by Citizen to run away from zombies.  I chose it to be virtual function because only some actors are zombies. By default, this function returns false. |
| *Virtual bool isFlameBlockable()* |
| Return true if the actor can block Flame.. This will be used by Wall and Exit to block Flame by Penelope.  I chose it to be virtual because some actors can block Flame. By default, this function returns false. |
| *Virtual bool isEnvironment()*  *Virtual bool isImmortal()*  *Virtual bool isCreature()*  *Virtual bool isGoodie()* |
| It turns out I don’t need to use these functions for the game. But I will keep them there in case I will need them in the newer version of the game. |
| *Virtual void getSick()* |
| This function is used by Penelope and Citizen to set their infection state be true and start the counting.  I chose it to be virtual function because only some actors need to use it. By default, this function does nothing. |
| *Virtual void exit()* |
| This function is used by Penelope and Citizen to exit the level.  I chose it to be virtual function because only some actors need to use it. By default, this function does nothing. |
| **Environment: Actor** |
| *Virtual bool isEnvironment()* |
| Always return true. |
| **ImmortalEnvironment: Environment** |
| *Virtual bool die()* |
| Always does nothing because ImmortalEnvironment (e.g. Wall )never dies. |
| *Virtual bool isAlive()* |
| Always return true because ImmportalEnvironment (e.g. Wall) never dies. |
| *Virtual bool isImmortal()* |
| Always return true. |
| **Wall: ImmortalEnvironment** |
| *Virtual void doSomething()* |
| Always do nothing because it’s just a wall. |
| *Virtual bool isFlameBlockable()* |
| Always return true because a wall can block Flame. |
| **Exit: ImmortalEnvironment** |
| *Virtual void doSomething()* |
| Check if it overlaps with Person. |
| *Virtual bool isOverlappable()* |
| Always return true because actors can overlap with exit. |
| *Virtual bool isFlameBlockable()* |
| Always return true. |
| **Pit: ImmortalEnvironment** |
| *Virtual void doSomething()* |
| Check if it overlaps with Creature. If it does, kill the Creature. |
| *Virtual bool isOverlappable()* |
| Always return true. |
| **MortalEnvironment: Environment** |
| *Virtual bool isOverlappable()* |
| Always return true, because all actors inherit from MortalEnvironment are overlappable. |
| **InflammableEnvironment: MortalEnvironment** |
| *Virtual bool isAlive()* |
| Return true if its life count is greater than 0. Otherwise return false. |
| *Virtual void die()* |
| Set its life count to be 0. |
| *Void decrementLife()* |
| Decrement its life count by 1. |
| **Flame: InflammableEnvironment** |
| *Virtual void doSomething()* |
| Check if it overlaps with actors. If it does, kill the actors. |
| **Vomit: InflammableEnvironment** |
| *Virtual void doSomething()* |
| Check if it overlaps with Person. If it does, they will get sick. |
| **FlammableEnvironment: MortalEnvironment** |
| *Virtual bool isFlammable()* |
| Always return true. |
| *Virtual void die()* |
| Set its state to be dead. |
| *Virtual bool isAlive()* |
| Return true if its m\_isAlive is true. Otherwise, return false. |
| **Landmine: FlammableEnvironment** |
| *Virtual void doSomething()* |
| Check if it overlaps with actors. If it does, explode. |
| *Void updateSafetyTicks()* |
| Decrement its safety tick until 0. |
| *Void explode()* |
| Introduce flame objects to its surroundings. Also a pit at its current location. |
| *Bool isActive()* |
| Return true if the safety ticks is 0. Otherwise return false. |
| **Goodie: FlammableEnvironment** |
| *Virtual bool isGoodie()* |
| Always return true. |
| **VaccineGoodie: Goodie** |
| *Virtual void doSomething()* |
| Check if it overlaps with Penelope. If it does, Penelope will eat this goodie. |
| **GasCanGoodie: Goodie** |
| *Virtual void doSomething()* |
| Check if it overlaps with Penelope. If it does, Penelope will eat this goodie. |
| **LandmineGoodie: Goodie** |
| *Virtual void doSomething()* |
| Check if it overlaps with Penelope. If it does, Penelope will eat this goodie. |
| **Creature: Actor** |
| *Virtual bool isCreature()* |
| Always return true. |
| *Virtual bool isFlammable()* |
| Always return true because all creatures can be killed by Flame. |
| *Virtual bool isAlive()* |
| Return true if its m\_isAlive is true. Otherwise return false. |
| *Virtual bool die()* |
| Set its state to be dead. |
| *Virtual double getStep() = 0* |
| This function returns number of pixels the creature can move every tick.  I chose it to be pure virtual because all creatures will implement this function. |
| *Virtual bool moveForward()* |
| Move forward to the direction its facing if there is nothing blocking it. |
| *Void setDead()* |
| Set its state to be dead. |
| *Void increment(int direction, double sizeX, double sizeY, double &x, double &y)* |
| A helper function for canMoveForward(). |
| *Bool canMoveForward(double &newX, double &newY)* |
| A helper function for moveForward() |
| **Zombie: Creature** |
| *Virtual void doSomething()* |
| Zombie will first check if it should be paralyzed.If not, check if it vomits. If not, it moves forward. |
| *Virtual bool isZombie()* |
| Always return true. |
| *Virtual void die()* |
| In addition to dying, it will notify the StudentWorld to increase the score and play the sound. |
| *Virtual int score() = 0* |
| This function returns the score for the world. When the zombie dies, the world will get this score.  This function is pure virtual function because every type of zombie will implement this function. |
| *Virtual double getStep()* |
| Returns the number of pixels the zombies move. |
| *Virtual bool moveForward()* |
| In addition to Creature’s moveForward, zombies will decrement its movement plan distance when it moves. |
| *Virtual void planMovement() = 0* |
| Set the movement plan distance and direction.  This function is pure virtual because different type of zombie has different way to plan its movement. |
| *Bool vomit()* |
| Introduce vomit object in front of it with a certain probablity. |
| *Void computeVomitCoordinates(double &vomitMinX, double &vomitMinY)* |
| Helper function for vomit to compute where the vomit object will be. |
| *Double getMovementPlanDistance()* |
| Getter function for movement plan distance. |
| *Void setMovementPlanDistace()* |
| Setter function for movement plan distance. |
| *Void decrementMovementPlanDistance()* |
| Decrement movement plan distance by 1. |
| *Bool isParalyzed()* |
| Return true is the zombie is paralyzed. Otherwise, return false. |
| **DumbZombie: Zombie** |
| *Virtual int score()* |
| Return the score for DumbZombie. |
| *Virtual void die()* |
| In addition to dying, it will drop vaccine with a certain probability. |
| *Virtual void planMovement()* |
| Plan a random movement. |
| *Void dropVaccineGoodie()* |
| Drop a vaccine goodie with a certain probability when it dies. |
| **SmartZombie: Zombie** |
| *Virtual int score()* |
| Return the score for SmartZombie. |
| *Virtual void planMovement()* |
| Attempts to plan the movement in such a way that it approaches to nearest person if it’s within a certain range. Otherwise, plan a random movement. |
| **Person: Creature** |
| *Virtual void doSomething()* |
| If a person is infected, and the infection count reaches a certain threshold, the person will mutate. For penelope, it simply dies. For citizen, it dies and becomes a zombie. |
| *Virtual void getSick()* |
| Set the infection state be true. |
| *Virtual bool isPerson()* |
| Always return true. |
| *Int getNumInfectionCount()* |
| Return how long the person has been infected. |
| *Virtual void injectVaccine()* |
| Set the infection state be false, and reset infection count. |
| *Virtual bool isSmart() = 0* |
| injectVaccine() will work only if the person is smart.  This function is pure virtual because this will be implemented for both Penelope and Citizen. |
| *Virtual void mutate() = 0* |
| A person will mutate when the infection count reaches a certain threshold. For Penelope, it simply dies, but for citizens, they become zombies. |
| *Bool isInfected()* |
| Return true if infected by Zombie vomit. Otherwise, return false. |
| **Penelope: Person** |
| *Virtual void doSomething()* |
| In addition to Person’s doSomething(), Penelope will handle user input. |
| *Virtual void exit()* |
| Penelope will complete the level when it overlaps with exit. |
| *Virtual void die()* |
| Set its state be dead. |
| *Bool didExit()* |
| Return true if Penelope exited the level. |
| *Void getVaccineGoodie(VaccineGoodie &goodie)* |
| Eat the goodie and increment vaccine count by 1. |
| *Void getGasCanGoodie(GasCanGoodie &goodie)* |
| Eat the goodie and increment flame count by 1. |
| *Void getLandmineGoodie(LandmineGoodie &goodie)* |
| Eat the goodie and increment landmine count by 1. |
| *Int getNumLandmines()* |
| Return how many landmines it has. |
| *Int getNumFlames()* |
| Return how many flames it has. |
| *Int getNumVaccines()* |
| Return how many vaccine it has. |
| *Virtual double getStep()* |
| Return number of pixels it can move in one tick. |
| *Virtual void injectVaccine()* |
| In addition to Person’s injectVaccine(), it also decrement the number of vaccines it has. |
| *Virtual bool isSmart()* |
| Always return true. |
| *Virtual void mutate()* |
| Just call die() |
| *Void handleKeyAction()* |
| Handle all the input from keyboard. |
| *Void handleLeftKey()* |
| Helper function for handleKeyAction(). |
| *Void handleRightKey()* |
| Helper function for handleKeyAction(). |
| *Void handleUpKey()* |
| Helper function for handleKeyAction(). |
| *Void handleDownKey()* |
| Helper function for handleKeyAction(). |
| *Void handleSpaceKey()* |
| Helper function for handleKeyAction(). |
| *Void handleTabKey()* |
| Helper function for handleKeyAction(). |
| *Void handleEnterKey()* |
| Helper function for handleKeyAction(). |
| *Void eat(Goodie &goodie)* |
| Helper function for getVaccineGoodie, getGasCanGoodie, and getLandmineGoodie. |
| **Citizen: Person** |
| *Virtual void doSomething()* |
| Try to approach Penelope. Otherwise try to run away from zombies. |
| *Virtual void exit()* |
| Exit the level. |
| *Virtual void die()* |
| Set its state be dead. |
| *Virtual void getSick()* |
| In addition to Person’s getSick(), it also notify the world to play sound. |
| *Virtual double getStep()* |
| Return number of pixels Citizen can move in one tick. |
| *Virtual bool isSmart()* |
| Always return false. |
| *Virtual void mutate()* |
| Dies and become a zombie. |
| *Bool approachPenelope(bool isZombieExist, double distaneToNearestZombie)* |
| Move towards Penelope. |
| *Bool approachPenelopeX(double penelopeMinX)* |
| Helper function for approachPenelope. |
| *Bool approachPenelopeY(double penelopeMinY)* |
| Helper function for approachPenelope. |
| *Bool approachPenelopeXY(double penelopeMinX, double penelopeMinY)* |
| Helper function for approachPenelope. |
| *Bool escapeFromZombies(bool isZombieExist, double distanceToNearestZombie)* |
| Move away from nearest zombie. |
| *Bool checkSafety(int direction, bool &isZombieExist, double &distance, double &newMinX, double &newMinY)* |
| Helper function for findSafestDirection. |
| *Bool findSafestDirection(double distanceToNearestZombie, int &newDirection, double &newMinX, double &newMinY)* |
| Helper function for escapeFromZombies. |
| **StudentWorld: GameWorld** |
| *Virtual int init()* |
| Initialize the game objects. |
| *Virtual int move()* |
| Each actor will do something. |
| *Virtual void cleanUp()* |
| Call destructors for each actor. |
| *Void handleVomit(Vomit &vomit)* |
| Check if there is any actor overlapping with vomit. |
| *Void handleVaccineGoodie(VaccineGoodie &goodie)* |
| Check if Penelope overlaps with vaccine goodie. |
| *Void handleLandmineGoodie(LandmineGoodie&goodie)* |
| Check if Penelope overlaps with landmine goodie. |
| *Void handlePit(Pit &pit)* |
| Check if there is any actor overlapping with pit. |
| *Void handleGasCanGoodie(GasCanGoodie &goodie)* |
| Check if Penelope overlaps with gas can goodie. |
| *Void handleLandmineExplosion(Landmine &landmine)* |
| Introduce Flame objects around landmine. Also introducing a pit at the center. |
| *Void handleFlame(Flame &flame)* |
| Check if flame overlaps with any actor. |
| *Void handleExit(Exit &exit)* |
| Check if exit overlaps with citizens or Penelope. |
| *Bool handleCanMoveTo(Actor &actor, double minX, double minY)* |
| Check if an actor can move to (minX, minY) |
| *Bool handleZombieVomit(Zombie &zombie, double vomitMinX, double vomitMinY)* |
| Introduce vomit with certain probability. |
| *Void getNearestPersonCoordinates(double x, double y, double &distance, double &personX, double &personY)* |
| Get the location of a person nearest to (x, y) |
| *Void handleCitizenMutation(Citizen &citizen)* |
| Introduce a smart zombie at the location of citizen. |
| *Void getPenelopeLocation(double &minX, double &minY)* |
| Get the location of Penelope. |
| *Double getDistanceToPenelope(double minX, double minY)* |
| Get the distance to Penelope from (minX, minY) |
| *bool getDistanceToNearestZombie(double minX, double minY, double &distance)* |
| Get distance to zombie nearest to (minX, minY) |
| *bool hasOverlapWithActorsOrPlayer(Actor &actor)* |
| Check if an actor overlaps with remaining actors or Penelope. |
| *bool hasOverlapWithActorsOrPlayer(double minX, double minY)* |
| Check if an object at (minX, minY) will overlap with any actor or Penelope. |
| *bool hasOverlapWithFlameBlockable(double minX, double minY)* |
| Check if an object at (minX, minY) has overlap with an actor whose isFlameBlockable() returns true. |
| *void generateFlames(double minX, double minY, int direction)* |
| Generate 3 flames from (minX, minY) to a given direction. |
| *void generateLandmine(double minX, double minY)* |
| Generate a landmine at (minX, minY). |
| *void decrementNumCitizens()* |
| Decrement number of citizens when a citizen dies. |
| *void dropVaccineGoodie(double minX, double minY)* |
| Drop vaccine goodie at (minX, minY) |
| *bool probability(double probability)* |
| Return true with given probability. |
| *bool loadLevelFile(Level &level, int &status)* |
| Load level file. |
| *void handleMazeEntry(Level &level, int levelX, int levelY)* |
| Generate actors. |
| *bool intersect(double minX, double minY, Actor const &other)* |
| Return true if object at (minX, minY) and actor intersects. |
| *bool overlap(double minX, double minY, Actor const &other)* |
| Return true if object at (minX, minY) and actor overlaps. |
| *void removeDeadActors()* |
| Destruct actors if they are dead. |
| *void updateDisplayText()* |
| Update the game information on the top. |

**A list of all functionality that you failed to finish as well as known bugs in your classes, e.g. “I didn’t implement the Flame class.” or “My smart zombie doesn’t work correctly yet so I treat it like a dumb zombie right now.”**

I finished all the functionalities as far as I know.

**A list of other design decisions and assumptions you made; e.g., “It was not specified what to do in situation X, so this is what I decided to do.”**

|  |
| --- |
| *“After exactly two ticks from its creation, the flame must set its state to dead so it can be destroyed and removed from the level by StudentWorld object”* |
| The definition of “after exactly two ticks from its creation” is too vague. During the second tick, it is not clear if the flame should do what it does or not. I decided not to do anything on the second tick. |
| *Landmine Flame Direction* |
| Landmine produces 9 flame objects when it explodes. It is not clear what the direction of the flame in the center should be. I made it have direction of right. |
| *Play sound when zombie attempts to vomit on a person* |
| Zombie vomits with a certain probability. It is not clear whether it should make a sound when it fails to vomit. I decided to have a vomit sound even when it fails to vomit. |

**A description of how you tested each of your classes (1-2 paragraphs per class).**

|  |
| --- |
| **Wall** |
| I checked if the Penelope will be blocked by the walls when it attempts to pass the walls. |
| **Exit** |
| I checked if Penelope and citizens can exit. |
| **Pit** |
| I checked if Penelope, citizens, and zombies die when it overlaps with pit objects. |
| **Flame** |
| I checked if Penelope, citizens, zombies, and goodies die when it overlaps with flame objects. I also checked if flame objects will be blocked by walls. |
| **Vomit** |
| I checked if Penelope and Citizens can be infected by Vomit and die eventually. |
| **Landmine** |
| I checked if the landmine will explode when an actor steps on it. I also checked if the explosion can be triggered by Flame. |
| **VaccineGoodie** |
| I checked if vaccine count increases when Penelope eats a vaccine goodie. |
| **GasCanGoodie** |
| I checked if flame count increases when Penelope eats a gas can goodie. |
| **LandmineGoodie** |
| I checked if landmine count increases when Penelope eats a landmine goodie. |
| **DumbZombie** |
| I checked if Penelope can be killed by DumbZombie. |
| **SmartZombie** |
| I checked if SmartZombie will chase Penelope and Citizens by replacing all DumbZombie with SmartZombie. It was a nightmare. |
| **Penelope** |
| I checked if Penelope is working by completing the game. |
| **Citizen** |
| I checked if Citizen will chase Penelope and run away from zombies. |
| **StudentWorld** |
| I checked if the game information on the top is displayed correctly. |