Dylan Lee

205 300 889

Project 3 Report

1. High-level descriptions of public member functions

* Actor

Actor(int startingHealth, int imageID, double startX, double startY, Direction dir = 0, int depth = 0, StudentWorld\* world = nullptr); - Passes all arguments except world to its base class, GraphObject. Stores world so that its derived classes can access its world’s public methods

double health() const; - Returns the Actor’s remaining health.

virtual void increaseHealth(int h); - Increases the Actor’s health by h up too 100. If health dips down to 0 or below, sets actor’s state to dead.

bool alive() const; - Return’s the Actor’s current state (true for alive, false for dead).

void setAlive(bool pAlive); - Sets the Actor’s current state (true for alive, false for dead).

StudentWorld\* getWorld(); - Returns a pointer to the Actor’s world.

virtual void doSomething() = 0; - Is called by StudentWorld’s move method for every Actor, every tick. Each Actor does something unique, so is set to virtual.

virtual bool damageable() = 0; - Returns whether or not the derived class can be damaged by flames or projectiles.

virtual bool blocks() = 0; - Returns whether or not the derived class blocks movement of Actors.

virtual bool edible() = 0; - Returns whether or not the derived class is edible by bacteria.

virtual bool noSpawnOverlap() = 0; - Returns whether or not the derived class can should not overlap with other classes when instantiated

virtual bool enemy() = 0; - Returns whether or no the derived class is an enemy, or spawns enemies. Used to check whether the player has completed the level.

* + Socrates

Socrates(double startX, double startY, StudentWorld\* world); - Initializes the player’s position, and gives Socrates a starting health of 100, his imageID, a Direction of 0, a depth of 0, 20 sprays, 5 flame charges, and an initial radial position of 180 degrees.

int getFlameCharges() const; - Returns the amount of flame charges Socrates currently has.

void addFlameCharges(int n); - Adds n flame charges to Socrates’s stockpile.

int getSprayCharges() const; - Returns the amount of sprays Socrates currently has.

virtual void doSomething(); - If Socrates isn’t alive, nothing happens. If he is, though, a key press will be taken from StudentWorld. If space is pressed, a spray object will be fired with its accompanying sound. If enter is pressed, 16 flame objects will extend out from Socrates in all directions. If a left or right key is pressed, Socrates will move counter-clockwise or clockwise, respectively. If no key is pressed, a spray will be added to Socrates’s count, up to 20.

virtual bool damageable(); - Returns false, as Socrates cannot be damaged by his own sprays and flames.

virtual bool blocks(); - Returns false, as Socrates does not block the movement of other Actors.

virtual bool edible(); - Returns false, as Socrates cannot be eaten.

virtual bool noSpawnOverlap(); - Returns false, as Socrates does not have to worry about spawn overlap.

virtual bool enemy(); - Returns false, as Socrates, being the player, is not an enemy.

* + DirtPile

DirtPile(double x, double y); - Creates a DirtPile object at position (x,y) with a 0 starting health, an imageID for dirt, a 0 degree direction, and a depth of 1.

virtual void doSomething(); - Returns immediately, as a DirtPile does nothing.

virtual bool damageable(); - Returns true, as a DirtPile can be destroyed with sprays or flames.

virtual bool blocks(); - Returns true, as a DirtPile can block other Actors.

virtual bool edible(); - Returns false, as a DirtPile cannot be eaten.

virtual bool noSpawnOverlap(); - Returns false, as a DirtPile can overlap with other DirtPiles.

virtual bool enemy(); - Returns false, as a DirtPile is not an enemy to the player.

* + Food

Food(double x, double y); - Creates a food object at position (x,y) with a 0 starting health, imageID for food, a direction of 90 degrees, and a depth of 1.

virtual void doSomething(); - Returns immediately, as Food does nothing.

virtual bool damageable(); - Returns false, as Food cannot be damaged.

virtual bool blocks(); - Returns false, as Food doesn’t block other actors.

virtual bool edible(); - Returns true, as Food can be eaten by Bacteria Actors.

virtual bool noSpawnOverlap(); - Returns true, as Food shouldn’t overlap with other fixed objects such as DirtPiles and Pits.

virtual bool enemy(); - Returns false, as Food is not an enemy.

* + Pit

Pit(double x, double y, StudentWorld\* world); - Creates a Pit object at position (x,y) with 0 starting health, an imageID for a Pit, a Direction of 0, and a depth of 1. One Pit can spawn 5 RegularSalmonella, 3 AggressiveSalmonella, and 2 EColi.

virtual void doSomething(); - If no more Bacteria are left to be released, set alive to false. If there are Bacteria left, there is a 1 in 50 chance one will be spawned. Of the Bacteria left, each has an equal chance of being spawned.

virtual bool damageable(); - Returns false, as Pits are not damageable.

virtual bool blocks(); - Returns false, as Pits do not block other Actors’ movement.

virtual bool edible(); - Returns false, as Pits cannot be eaten.

virtual bool noSpawnOverlap(); - Returns true, as Pits should not overlap with other objects when spawned.

virtual bool enemy(); - Returns true, as the level can only end when all Bacteria and Pits are removed from the level

* + Projectile

Projectile(double x, double y, Direction d, int imgID, int maxDistance, int damage, StudentWorld\* world); - Creates a Projectile object at position (x,y), which travels a maximum of maxDistance pixels, and inflicts damage when it collides with an enemy. Initializes distance travelled to zero

int getDistance() const; - Returns the distance the Projectile has travelled thus far.

void increaseDistance(int d); - Increases the distance the Projectile has travelled.

int getMaxDistance() const; - Returns the maximum distance the Projectile can travel.

int getDamage() const; - Returns the amount of damage the Projectile inflicts on enemies.

virtual void doSomething(); - Returns immediately if the Projectile is not alive. If it is, it checks to see if it collided with another Actor that is damageable, in which case it inflicts damage and is removed. If none of those conditions are met, it continues to travel along its current trajectory.

virtual bool damageable(); - Returns false, as a Projectile cannot be damaged. It is “killed” when it hits an Actor that is damageable.

virtual bool blocks(); - Returns false, as Projectiles don’t block the movement of other Actors.

virtual bool edible(); - Returns false, as Projectiles cannot be eaten.

virtual bool noSpawnOverlap(); - Returns false, as Projectiles’ spawn location isn’t restricted.

virtual bool enemy(); - Returns false, as Projectiles are not enemies to the player.

* + - Flame

Flame(double x, double y, Direction d, StudentWorld\* world); - Creates a Flame Projectile at position (x,y) with a flame imageID, and a max movement distance of 32 pixels, which does 5 damage to enemies.

* + - Spray

Spray(double x, double y, Direction d, StudentWorld\* world); - Creates a Spray Projectile at position (x,y) with a spray imageID, and a max movement distance of 112 pixels, which does 2 damage to enemies.

* + Pickup

Pickup(int imageID, double x, double y, StudentWorld\* world); - Creates a pickup at position (x,y). Sets age to zero and lifetime to a random number from 50 to 300 minus 10 times the level minus 1.

virtual void doSomething(); - Returns immediately if the Pickup is not alive. If it is, it increments age and checks if it overlaps with Socrates. If the Pickup does overlap with Socrates, the effectOnSocrates() function is called. If the Pickup’s age equals its lifetime, its state is set to dead.

virtual void effectOnSocrates() = 0; - Does something to Socrates. Declared virtual because each Pickup does something different to Socrates.

virtual bool damageable(); - Returns true because Pickups can be damaged by Projectiles.

virtual bool blocks(); - Returns false because Pickups don’t block other Actors’ movements.

virtual bool edible(); - Returns false because Pickups cannot be eaten.

virtual bool noSpawnOverlap(); - Returns false because Pickups can overlap with other Actors on spawn.

virtual bool enemy(); - Returns false because Pickups don’t need to be destroyed to complete the level

* + - HealthGoodie

HealthGoodie(double x, double y, StudentWorld\* world); - Creates a HealthGoodie at position (x,y) with a health goodie imageID.

virtual void effectOnSocrates(); - Increases score by 250, plays a sound, and refills Socrates’s health.

* + - FlamethrowerGoodie

FlamethrowerGoodie(double x, double y, StudentWorld\* world); - Creates a FlamethrowerGoodie at position (x,y) with a flame thrower goodie imageID.

virtual void effectOnSocrates(); - Increases score by 300, plays a sound, and gives Socrates five more flamethrower charges.

* + - ExtraLifeGoodie

ExtraLifeGoodie(double x, double y, StudentWorld\* world); - Creates an ExtraLifeGoodie at position (x,y) with an extra life goodie imageID.

virtual void effectOnSocrates(); - Increases score by 500, plays a sound, and increases Socrates’s lives by one.

* + - Fungus

Fungus(double x, double y, StudentWorld\* world); - Creates a Fungus at position (x,y) with a fungus imageID.

virtual void effectOnSocrates(); - Decreases score by 50, and Socrates’s health by 20 points.

* + Bacteria

Bacteria(int startingHealth, int imageID, double x, double y, int damage, int movementDistance, StudentWorld\* world); - Creates a Bacteria at position (x,y), that does damage to the player when it collides with them. It initializes its movement plan distance and number of food eaten to zero.

virtual void doSomething(); - Returns immediately if Bacteria is no longer alive. If the Bacteria is alive, it may or may do something extra, which it keeps track of. Then, it will check if it overlaps with Socrates, in which case it will damage them. Else, if it has eaten three food, it will create a new Bacteria of its type near itself. Else, if it overlaps with food, it will eaten it. If the doSomethingExtra() method returned true, it will return. If doSomethingExtra() returned false, it will attempt to move.

virtual bool doSomethingExtra() = 0; - Aggressive Salmonella will attempt to move towards Socrates in doSomething(), so it will return true. All other types of Bacteria’s doSomethingExtra() functions will return false.

virtual void addBacteria(double x, double y) = 0; - Creates a Bacteria of the derived class’s type.

virtual void increaseHealth(int h); - Changes the health of the Bacteria by h, playing appropriate sounds and checking for death. If the Bacteria dies, there is a 50% chance a food will be spawned at their location.

virtual void playHurtSound() = 0; - Plays the bacteria’s hurt sound. This is different for Salmonella and EColi.

virtual void playDieSound() = 0; - Plays the bacteria’s death sound. This is different for Salmonella and EColi.

virtual bool damageable(); - Returns true, as Bacteria can be damaged by Projectiles.

virtual bool blocks(); - Returns false, as Bacteria don’t block other actors.

virtual bool edible(); - Returns false, as Bacteria can’t be eaten.

virtual bool noSpawnOverlap(); - Returns false, as Bacteria can spawn on top of each other.

virtual bool enemy(); - Returns true, as Bacteria are enemies, and must be destroyed for the level to complete.

virtual void attemptMovement() = 0; - The Bacteria tries to move in either its current or a new direction. Is different for Salmonella and EColi.

bool checkMovement(int n); - Checks the spot n pixels in front of the Bacteria for either DirtPiles or the edge of the accessible area.

void resetDirection(); - Sets the Bacteria’s Direction to a random angle between 0 and 359. Also resets the movement plan distance to 10.

int getMovementPlanDistance() const; - Returns the number of ticks the Bacteria plans to continue moving in its current direction.

void resetMovementPlanDistance(); - Resets the Bacteria’s movement plan distance to 10.

void decMovementPlanDistance(); - Decrements the Bacteria’s movement plan distance by 1.

int getFoodEaten() const; - Returns the number of Food objects the Bacteria has eaten.

void resetFoodEaten(); - Resets the number of Food objects the Bacteria has eaten to zero.

void incFoodEaten(); Increments the number of Food objects the Bacteria has eaten by 1.

int getDamage() const; - Gets the amount of damage the Bacteria does to the player on a collision.

int getMovementDistance() const; - Returns the amount of pixels the Bacteria moves per tick.

* + - Salmonella

Salmonella(int startingHealth, double x, double y, int damage, StudentWorld\* world); - Creates a Salmonella object at position (x,y) with a salmonella imageID, and a movement distance of 3.

virtual void playHurtSound(); - Plays the salmonella’s hurt sound.

virtual void playDieSound(); - Plays the salmonella’s die sound.

virtual void attemptMovement(); - Attempts to move in the Bacteria’s current direction if it’s movement plan distance is greater than zero. If it is blocked from moving in its current direction, it will reset it. If the Bacteria’s movement plan distance is zero, it will attempt to find the closest food within 128 pixels, if it can’t it will reset its direction and movement plan. If it can find, it will try to move towards it. If it can’t move towards the food, it will reset its direction and movement plan distance.

* + - * RegularSalmonella

RegularSalmonella(double x, double y, StudentWorld\* world); - Creates a RegularSalmonella object at position (x,y), with starting health 4, and 1 damage.

virtual bool doSomethingExtra(); - Returns false, as the RegularSalmonella doesn’t do anything extra.

virtual void addBacteria(double x, double y); - Adds a new RegularSalmonella at position (x,y).

* + - * AggressiveSalmonella

AggressiveSalmonella(double x, double y, StudentWorld\* world); - Creates an AggressiveSalmonella object at position (x,y), with starting health 10, and damage 2.

virtual bool doSomethingExtra(); - If Socrates is within 72 pixels, it will set the AggressiveSalmonella’s direction towards him and move in that direction. If it can’t move in that direction or Socrates is not within range, it will stay still.

virtual void addBacteria(double x, double y); - Adds a new AggressiveSalmonella at position (x,y).

* + - EColi

EColi(double x, double y, StudentWorld\* world); - Creates an EColi object at position (x,y), with starting health 5, damage 4, and movement distance 2.

virtual bool doSomethingExtra(); - Returns false, as EColi doesn’t do anything extra.

virtual void playHurtSound(); - Plays EColi’s hurt sound.

virtual void playDieSound(); - Plays EColi’s death sound.

virtual void attemptMovement(); - Will attempt to move towards Socrates, if it can’t it will add 10 to its direction up to 10 times. If none of those angles works, the EColi will remain still.

virtual void addBacteria(double x, double y); - Adds a new EColi object at position (x,y).

* StudentWorld

StudentWorld(std::string assetPath); - Initializes the Socrates pointer to nullptr and seeds rand.

virtual int init(); - Dynamically allocates Socrates, as well as Pits, Food, and DirtPiles, the number of which are determined based on the level and rand number generation. Pushes all actors except Socrates into a list.

virtual int move(); - Iterates through the Actor linked list, calling their doSomething() method unless they are not alive. If the player is dead, it decrements their lives and returns from the level. If all the enemies have been cleared, the level completes. Also iterates through the Actors list again, removing any dead Actors. Goodies are potentially spawned and the HUD is updated.

virtual void cleanUp(); - Deallocates the Socrates pointer and all the pointers within the Actor list.

void addActor(Actor\* a); - Pushes a new actor to the front of the Actor list.

void addGoodies(); - Uses a number random generator to potentially generate goodies/fungi.

void addCharges(int n); - Adds n charges to Socrates’s inventory.

void increaseSocratesHealth(int h); - Increases Socrates’s health by h.

bool checkProjectileCollision(Projectile\* src); - Returns whether or not src is within SPRITE\_WIDTH pixels of a damageable Actor. If an Actor overlaps and its damageable, the Projectile does damage and is removed.

bool checkSocratesCollision(Actor\* src); - Returns whether or not Socrates is within SPRITE\_WIDTH pixels of src.

bool checkSocratesWithinRange(Actor\* src, double range); - Returns whether or not see if Socrates is within range pixels of src.

bool checkDirtPileCollision(Actor\* src); - Returns whether or not src is within SPRITE\_WIDTH / 2 pixels of a dirt pile.

bool checkFoodOverlap(Actor\* src); - Returns whether or not src is within SPRITE\_WIDTH pixels of a Food object. If it is, sets the Food object’s state to dead.

bool checkSpawnOverlap(double x, double y); - Returns whether or not any Actors who are not supposed to spawn overlap are within SPRITE\_WIDTH pixels of position (x,y).

bool getClosestFood(Actor\* src, Direction &d); - Returns true if a Food object is within 128 pixels of src. Sets d to the direction of the closet Food object, if one exists.

int getSocratesDirection(Actor\* src); - Returns the direction Socrates is in relation to src.

~StudentWorld(); - Calls cleanup to deallocate all Actors.

1. List of unfinished functionalities and known bugs
   * There are no known unfinished functionalities
   * When the program is running especially fast, holding down the spacebar when the spray count is at zero will continue to fire sprays as it replenishes them. Presumably, this is because the game is running faster than input can be collected, as it also happens in the sample program.
   * The sprays in the sample program seem to travel a little further than my version of the program, but my version does adhere to the specification’s required 112 pixel travel distance for the sprays.
2. List of design decisions and assumptions
   * The spec did not specify whether or not the Salmonella’s movement plan distance should be reset if they were successful in moving towards the nearest food. Because of this, in the Salmonella class’s attemptMovement function, if moving towards a Food object is successful, the Salmonella does not call its resetMovementPlanDistance() function on that tick.
3. Test Description
   * Socrates
     + I added Socrates to an empty level and made sure his controls worked, that he moved around the petri dish, all the while looking at the middle. I also tested his sprays and flames, making sure that when his flames and sprays ran out, he could no longer produce new ones. A lot of it came down to comparing behavior between the sample program’s version of Socrates and mine.
   * DirtPile
     + I populated an empty level with DirtPiles, making sure the configuration changed every time I booted up the game.
   * Food
     + Like the DirtPiles, I populated empty levels with Food, making sure they changed configurations every time, and didn’t overlap with DirtPiles or Pits.
   * Pit
     + I created a few levels and made sure the Pits were spawning in different places throughout the level. At one point, I noticed that the Pits were spawning Bacteria infinitely, instead of the customary 5, 3, and 2. To remedy this, I printed out the number of Bacteria the Pits had left after each spawn, and noticed that they were going into the negative. I changed my code to account for when one number was zero while the others weren’t, and then fixed my code so that Bacteria weren’t spawned if the Pit should not have had any more left.
   * Flame
     + I created a level that just contained Socrates and DirtPiles, and made sure that Flames destroyed them, while also only damaging one DirtPile per flame. I also referenced the sample program to make sure my flame configuration looked similar and disappeared after travelling a short distance. I also tested the flames against enemies, making sure that they damaged the Bacteria right after the two objects collided. I also tested them against goodies.
   * Spray
     + I created a level that just contained Socrates and DirtPiles, and made sure that Sprays destroyed them, one at a time. I also tested them against enemies, making sure that that destroyed RegularSalmonella in 2 hits, AggressiveSalmonella in 5, and EColi in 3.
   * HealthGoodie
     + I created a level with preset HealthGoodies and made sure that they restored me all the way up, but not over, 100 health and increased my score by 250 points. I also made sure they could be destroyed by Flames, but not Bacteria.
   * FlamethrowerGoodie
     + I created a level with preset FlamethrowerGoodies and made sure that they gifted me five more flame charges, with no maximum, and gave me 300 points. I also made sure they could be destroyed by Flames, but not Bacteria.
   * ExtraLifeGoodie
     + I created a level with preset FlamethrowerGoodies and made sure that they gifted me one extra life, with no maximum, and gave me 500 points. I made sure that the extra life was added and died three more times to see that it was counted. I also made sure they could be destroyed by Flames, but not Bacteria.
   * Fungus
     + I created a level with preset Fungus objects and made sure that they decreased my health by exactly 20 and decreased my score by 50 points. I also made sure that they could be destroyed by Flames, but not Bacteria.
   * RegularSalmonella
     + I added Socrates, RegularSalmonella, Food, and DirtPiles to a level and observed how the RegularSalmonella behaved, paying attention to its movement through the DirtPiles, and its propensity to seek the closest Food. I also made sure they didn’t deliberately attack Socrates.
   * AggressiveSalmonella
     + I added Socrates, RegularSalmonella, Food, and DirtPiles to a level and observed how the RegularSalmonella behaved, paying attention to its movement through the DirtPiles, and its propensity to seek the closest Food. I also observed how when Socrates got close, it actively pursued him.
   * EColi
     + I added Socrates, EColi, Food, and DirtPiles to a level and observed how the RegularSalmonella behaved, paying attention to its movement through the DirtPiles, and how it sought out Socrates, but could get stuck behind DirtPiles.