1. Notes: all my functions were public. Projectile, Actor, Goodie and Zombie classes were abstract base classes whose only pure virtual function was the void doSomething either directly from Actor class itself or inherited, and these functions do nothing by definition since their bodies are empty. But each derived class had different doSomething functionality, so I decided to define each myself. The constructors with just curly braces do nothing in main body, since there was no reason for them to do anything besides utilizing initializer list to call the parametrized constructors of immediate base class. Destructors did nothing to delete themselves because I had cleanup from StudentWorld do all the deletion; all they did was to enable deallocation once those objects went out of scope. Also, all my classes had virtual destructors to prevent memory leak from happening. Also, I will not give pseudocode for trivial functions since those short line functions are self-explanatory.

Also, human is another way of saying either Penelope or citizen. And whenever I say kill zombie or citizen, I indirectly imply adding/subtracting proper score value as described by spec.

Some functions did not need virtual, but I still added them as either a sanity check or to test virtual functionality. For ones that didn’t, I didn’t add because technically the program would still compile without the word (for StudentWorld)

I also made most of my functions virtual because just in case my pointer to studentWorld was unable to see any of these functions. For studentWorld, I defined them all there because no class is derived from StudentWorld and to enable my actor class to access information on member variables, such as my list of pointers, without making the variables public.

Below is pseudocode for my functions

**StudentWorld File:**

StudentWorld(std::string assetPath);

Used initializer list to initialize penelope to nullptr and set level finished to false

virtual int init();

Setup level using ostringstream

If no level found or level greater than 99, return player won

Or If level improperly formatted, return level error

Or if level load success, convert each character in text file to respective objects and dynamically allocate and store to a list of actor pointers

Return game continue

virtual int move();

Assume level is not finished

If Penelope or other actors are alive, tell her or them to do something

If Penelope is dead, play player died sound, decrease life by 1, and return player died

return level finished if level is finished

delete all dead actors and erase slot from list

use ostringstream to keep track of scores, level, lives, etc.

return continue game

virtual void cleanUp();

delete all actors in list and erase each slot from the list

delete Penelope and reset her to nullpointer

virtual bool interference(double x, double y) const; //used to access beginning of m\_actors for Penelope

//This was made in consideration for part 1 only, but since it still worked, I decided to keep it and have a //second function that overloaded it

Repeatedly,

If Penelope’s graphic intersects with that of another actor that can block, then block Penelope movement

virtual bool overlap(double x1, double y1); //used by exit to get Penelope/citizen to leave level

repeatedly,

If citizen found in list and it’s within Euclidean distance of 10, then let citizen leave level and increase user’s score

virtual bool interference(double x1, double y1, double x2, double y2) const; //used by other actors themselves

repeatedly,

If zombie is calling this and we find matching zombies, ignore

Or If citizen is calling this and we find matching citizen, ignore

Or If their graphic blocks intersected because of a moving actor, then block the movement

Or if caller coordinates intersect with those of Penelope, block movement

virtual bool overlap(double x1, double y1, double& x2, double& y2); //can used by flame, vomit, and pit

For each actor,

If zombie coordinates overlap with pit’s, then play sound, set it to dead, and increase score, and if dumb zombie killed, check to see if by chance vaccine can be added.

If citizen coordinates overlap, then kill citizen

If penelope’s coordinates overlap, then kill her

virtual void burn(double x1, double y1, double& x2, double& y2);

repeatedly,

if dumb or smart zombie overlaps, then kill it, and if dumb, then create new vaccine.

Or if citizen overlaps, then kill citizen

Or if goodie overlaps, then destroy it

Or if landmine overlaps, then set it to explode

Or if Penelope overlaps, then kill her

bool vomit(double x1, double y1, double& x2, double& y2);

with 1/3 chance,

return true if human overlaps with vomit coordinates

void infect(double x1, double y1, double& x2, double& y2);

repeatedly,

set citizen’s and penelope’s infect status to true if it’s still false if they overlap with vomit cooridinates

double minDist(double x1, double y1, double& x2, double& y2) const;

compare zombie’s distance to Penelope if passed in as parameter

or repeatedly,

compare closest citizen’s distance to zombie, and return smallest distance

void findHuman(double x1, double y1, double& x2, double& y2);

repeatedly,

find next citizen in list whose row or column coordinate matches zombie’s

virtual double distAway(double x, double y, double& zx, double& zy) const;

repeatedly,

find each zombie, and return minimum distance to citizen that calls this.

virtual void actorCounter(int& obj1, int& obj2) const; //used to count #zombies and citizens in level

repeatedly,

count total number of zombies and citizens in level

//void addActor(char actor, double x, double y, Direction dir);

void addActor(Actor\* p);

add actor to list

bool canaddVaccine(double x, double y);

repeatedly,

if Vaccine’s coordinates overlap with another object in list, return false

bool canadd(double x, double y) const;

repeatedly,

if wall or exit does not overlap with new flame’s coordinates, then return true

virtual void levelFinished() { m\_levelFinished = true; } //can be called

virtual bool getLevelFinished() const { return m\_levelFinished; }

Actor\* getPenelope() const { return m\_penelope; } //so citizen can access Penelope's coordinates

Return Penelope pointer

//StudentWorld's destructor

virtual ~StudentWorld();

call cleanup

**Actor File:**

Actor Class:

Actor(int imageID, double startX, double startY, Direction dir , int depth , StudentWorld\* sworld);

Call GraphObject’s parametrized constructor and initialized studentworld pointer, alive status, infectstatus, infect counter, overlap and exit traits, cankill trait and paralysis tracker

virtual void doSomething() = 0;

StudentWorld\* getWorld() { return m\_world; }

//part 2 stuff

Note: I use virtual because I assumed in general most shared a particular similar trait by default, and for ones that didn’t, I overwrote myself.

Also, most constructors just call immediate base class’s parametrized constructor

virtual bool checkliving() const { return m\_alive; }

All actors can check to see if alive; virtual since it is possible that walls or exits will never die

virtual bool canhurt() const { return true; } //only walls, pits and exits are immune

all actors can check if it can be killed. Only walls, exits and pits are immune, so they’re different.

virtual bool canoverlap() const { return m\_canoverlap; } //using euclidean distance

all actors can check if it can overlap with other objects. Some objects overlap, and some don’t.

virtual bool canexit() const { return m\_canexit; } //only Penelope and actors may pass through exits

all can check to see if it can exit. Most can’t besides humans.

virtual bool canKill() const { return m\_cankill; }

Only zombies can kill other people, while others cannot.

virtual bool checkInfection() const { return m\_infectStatus; }

In case non-human objects may be infected, I assumed they never will be.

virtual bool isSmart() const { return false; }

All actors are by default dumb, except for smart zombie.

virtual bool isFlammable() const { return true; } //flames cannot pass through exits or walls

All actors can be burned, except for exits and walls and pits and projectiles

virtual bool blockFlame() const { return false; }

same as above

virtual bool canexplode() const { return false; }

Only landmines can explode, but all others cannot.

virtual void beginexplosion() {}

Only one that explodes is landmines; others should do nothing

int infectCount() const { return m\_infectCount; }

all can check their infection counter

void resetOverlap() { m\_canoverlap = true; } //enable us to step on exits/goodies/pits/flame/mines

void resetCanexit() { m\_canexit = !m\_canexit; }

void resetCankill() { m\_cankill = !m\_cankill; }

void resetInfection() { m\_infectStatus = false, m\_infectCount = 0; } //enables vaccine to cure

void resetStatus() { m\_alive = false; } //no problem if called multiple times

void increaseInfect() { m\_infectCount++; }

void infectme() { m\_infectStatus = true; }

//paralysis implementation for citizens and zombies

bool isParalyzed() const { return m\_paralyzed; }

void resetParalysis() { m\_paralyzed = !m\_paralyzed; }

//Just to enable m\_penelope to access Penelope's version (hence, using virtual). These don't do anything special

virtual int flameCount() const { return -1; }

virtual int vaccineCount() const { return -1; }

virtual int mineCount() const { return -1; }

virtual void changeFlame(int n) { ; }

virtual void changeVaccine(int n) { ; }

virtual void changeMine(int n) { ; }

virtual ~Actor();

Penelope Class:

Penelope(double start\_x, double start\_y, StudentWorld\* sworld);

Call actor’s constructor and initialize

virtual void doSomething();

increase infect counter if infected

return if dead or infect counter reaches 500

take input from user for movement,

move up, down, left, or right if no blockage

fire flamethrower if flames left

use vaccine if she has at least one

plant a mine if she has one

virtual ~Penelope();

virtual bool canexit() const { return true; }

virtual int flameCount() const { return m\_flamecharges; }

virtual int vaccineCount() const { return m\_vaccines; }

virtual int mineCount() const { return m\_mines; }

virtual void changeFlame(int n) { m\_flamecharges += n; }

virtual void changeVaccine(int n) { m\_vaccines += n; }

virtual void changeMine(int n) { m\_mines += n; }

Wall Class:

Wall(double startX, double startY, StudentWorld\* sworld);

virtual void doSomething();

virtual ~Wall() {}

virtual bool canhurt() const { return false; } //although walls and exits can block flame, pits cannot

virtual bool isFlammable() const { return false; }

virtual bool blockFlame() const { return true; }

Exit Class:

Exit(double startX, double startY, StudentWorld\* sworld);

virtual void doSomething();

check if citizen and Penelope overlap,

set level finished to true if all citizens gone from level

virtual bool canhurt() const { return false; }

virtual bool isFlammable() const { return false; }

virtual ~Exit() {}

virtual bool blockFlame() const { return true; }

Citizen Class:

Citizen(double startX, double startY, StudentWorld\* sworld);

Reset permission to exit level to true, since false by default for actors besides penelope

virtual void doSomething();

return if dead or if paralysis tick is true

increase infection counter if infected

if infection counter is 500,

set alive to false and play zombie born sound, 70% dumb, 30% smart zombie

compare distances away from Penelope and closest zombie

if there’s no blockage of movement, change movement and direction for citizen if Penelope is closer and within 80-pixels so that citizen gets closer to her

or if there is at least 1 zombie within 80-pixels that is closer to citizen that Penelope,

pick a direction and move so that citizen increases distance away from closest zombie, or stay if its current position is farthest possible distance.

virtual ~Citizen() {}

Zombie Class:

Zombie(double startX, double startY, StudentWorld\* sworld);

virtual ~Zombie() {}

int getMovePlan() const { return m\_moveplan; }

void resetMove(int x) { m\_moveplan = x; } //reset moveplan to 0 or some randInt value

void changeMove(int x) { m\_moveplan += x; }

Dumb Zombie Class:

DumbZombie(double startX, double startY, StudentWorld\* sworld);

virtual void doSomething();

return if dead or paralyzed

find nearby human within its current direction

if human is nearby and falls within zombie’s vomit coordinates, introduce vomit and play sound

if moveplan is 0, set random number between 3 and 10 and update moveplan with it, and randomly reset direction

if zombie can move in its current direction without blockage, move and decrease moveplan by 1

or else reset moveplan to 0

virtual ~DumbZombie() {}

Smart Zombie Class

SmartZombie(double startX, double startY, StudentWorld\* sworld);

virtual void doSomething();

return if dead or paralyzed

find nearby human within its current direction

if human is nearby and falls within zombie’s vomit coordinates, introduce vomit and play sound

if moveplan is 0, set random number between 3 and 10 and update moveplan with it, and compare distance between closest citizen and Penelope.

Pursue the closer human, and change its direction so that it faces the desired human and move it so that the distance to human decreases

if zombie can move in its current direction without blockage, move and decrease moveplan by 1

or else reset moveplan to 0

virtual ~SmartZombie() {}

virtual bool isSmart() const { return true; }

Goodie Class:

Goodie(int imageID, double startX, double startY, StudentWorld\* sworld);

virtual ~Goodie() {}

VaccineGoodie Class:

VaccineGoodie(double startX, double startY, StudentWorld\* sworld);

virtual ~VaccineGoodie() {}

virtual void doSomething();

give Penelope 1 vaccine if she overlaps and +50 score

GasCanClass:

GasCanGoodie(double startX, double startY, StudentWorld\* sworld);

virtual ~GasCanGoodie() {}

virtual void doSomething();

give penelope 5 flame charges if she overlaps and +50 score

LandmineGoodie Class:

LandmineGoodie(double startX, double startY, StudentWorld\* sworld);

virtual ~LandmineGoodie() {}

virtual void doSomething();

give Penelope 2 landmines if she overlaps and +50 score

Landmine Class:

Landmine(double startX, double startY, StudentWorld\* sworld);

Enable overlapping

virtual ~Landmine() {}

virtual void doSomething();

return if not alive

decrease safety tick if greater than zero, or activate it if inactive

call beginexplosion() if there is overlap with any actor

virtual bool canexplode() const { return true; }

virtual void beginexplosion();

for each of eight slots around the center plus center itself,

create flame object if no wall/exit overlaps

then create pit at position of previous landmine

int safetyCount() const { return m\_safetytick; }

bool isActive() const { return m\_active; }

void activate() { m\_active = true; }

Pit Class:

Pit(double startX, double startY, StudentWorld\* sworld);

Reset overlap to true

virtual ~Pit() {}

virtual void doSomething();

check if any actor overlaps with it, and then set alive status to false

virtual bool canhurt() const { return false; }

virtual bool isFlammable() const { return false; }

pit should not be flammable

Projectile Class:

Projectile(int imageID, double startX, double startY, Direction dir, StudentWorld\* sworld);

Set overlap to true

virtual ~Projectile() {}

int tickCount() const { return m\_tickCount; }

virtual bool canhurt() const { return false; }

void increaseTick() { m\_tickCount++; }

virtual bool isFlammable() const { return false; }

flames and vomit should not be burnable

Flame Class:

Flame(double startX, double startY, Direction dir, StudentWorld\* sworld);

Set overlap to true

virtual ~Flame() {}

virtual void doSomething();

set any actor that overlaps with it to alive status of false

Vomit Class:

Vomit(double startX, double startY, Direction dir, StudentWorld\* sworld);

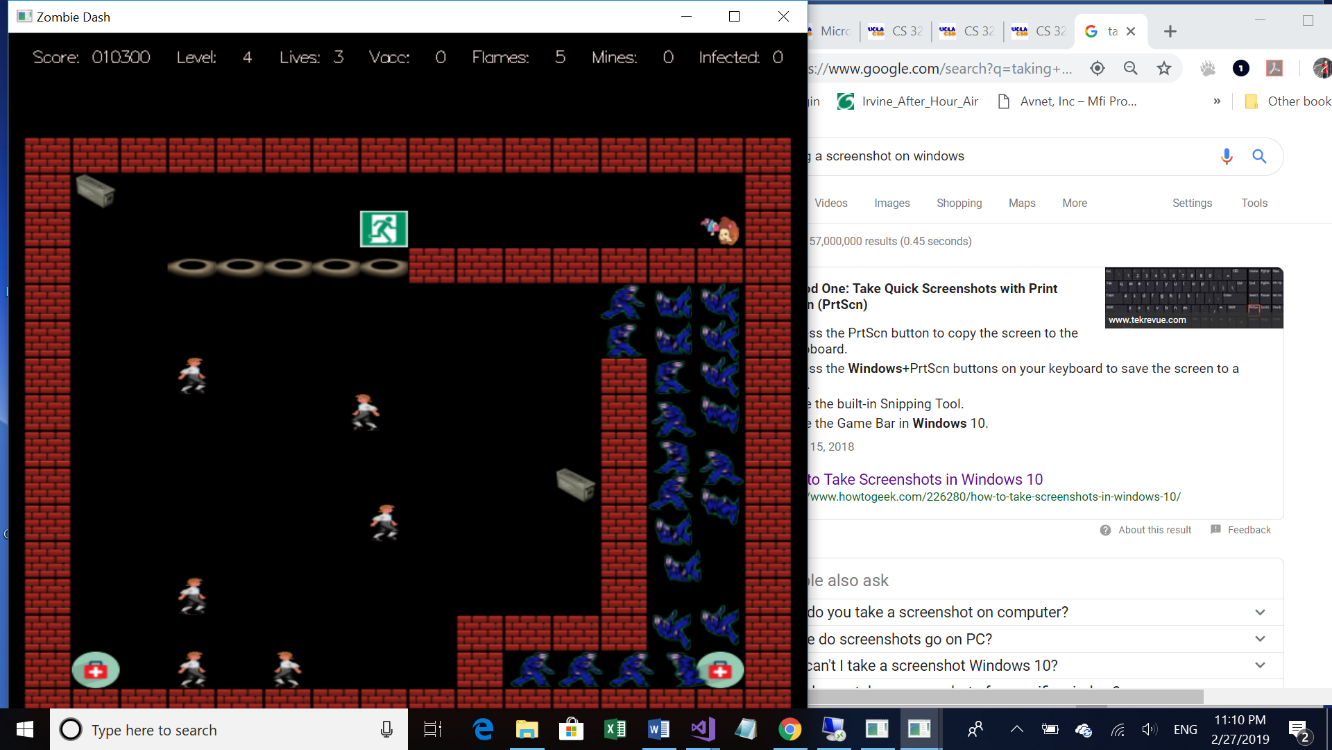
Set overlap to true

virtual ~Vomit() {}

virtual void doSomething();

infect any human that is not infected yet that overlaps with vomit

2. In level 4, when the smart zombies were cornering some citizens or Penelope, if a human gets caught in a corner, some of the zombies freeze up unless either I try to move elsewhere or burn some away, as shown in screenshot. 4 zombies directly below Penelope are completely frozen.



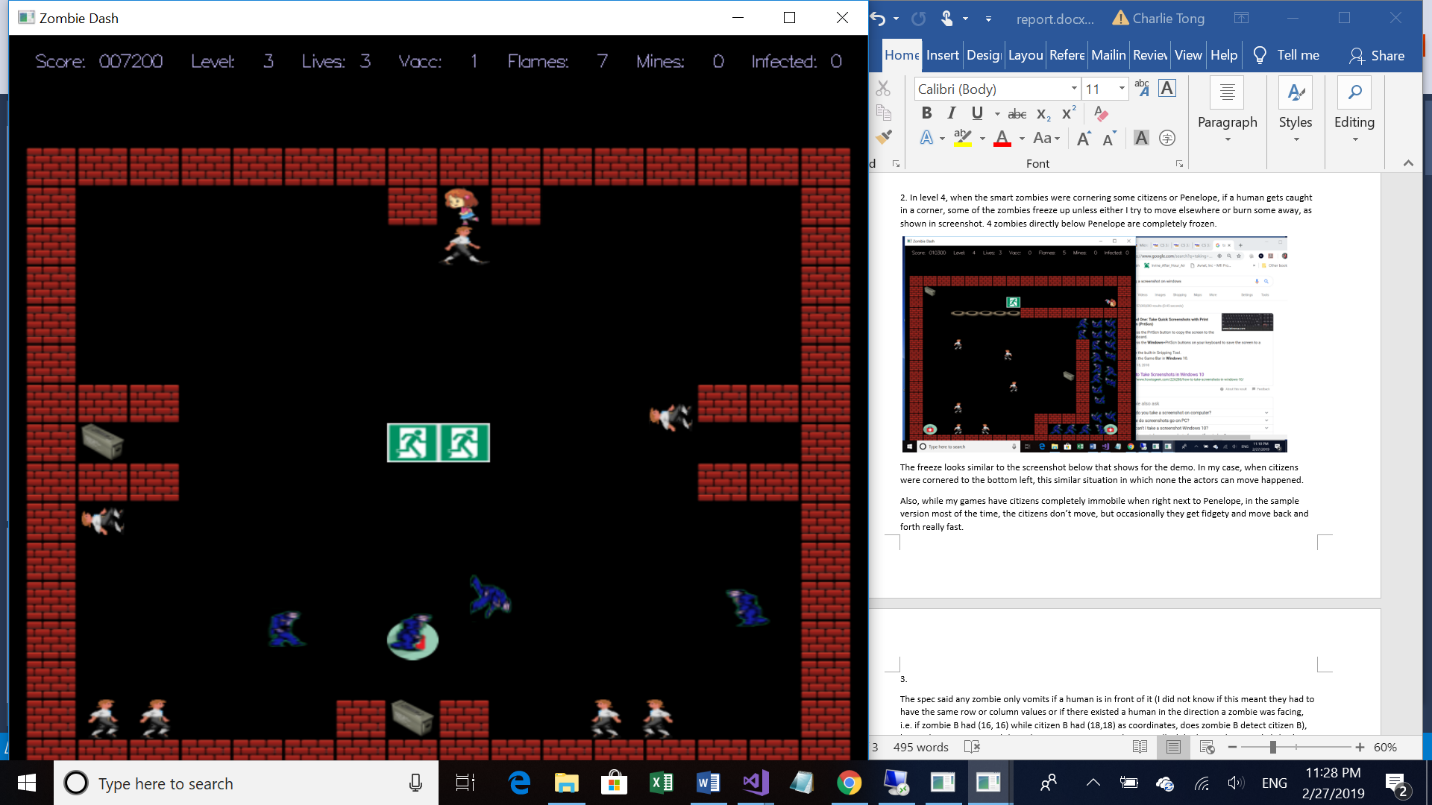
The freeze looks similar to the screenshot below that shows the demo. In my case, when citizens were cornered to the bottom left, this similar situation in which none the actors can move happened.

Also, while my games have citizens completely immobile when right next to Penelope, in the sample version most of the time, the citizens don’t move, but occasionally they get fidgety and move back and forth really fast.

3.

The spec said any zombie only vomits if a human is in front of it (I did not know if this meant they had to have the same row or column values or if there existed a human in the direction a zombie was facing, i.e. if zombie B had (16, 16) while citizen B had (18,18) as coordinates, does zombie B detect citizen B), but in the game, I noticed that when a citizen was just above a wall while the zombie was slightly above the zombie, the zombie still vomited on the citizen. What I ended up doing was to have zombie vomit even if a human was not exactly in the same row/column, provided that there was a human in front of zombie. For example, suppose citizen A had coordinates (16,16), while zombie A had (18, 18). I had zombie still vomit as long as citizen A was in the range of vomit coordinate.

The spec never specified what to do if Penelope gets caught in a corner, as shown, so I followed the demo and remained stuck unless I kill the interfering citizens myself.



I also made the assumption that if zombie sees a human nearby, it will call vomit. It does not check if the direction vomited is blocked since I assumed that Penelope and citizens will never overlap with a wall, which I believed was the only thing that could block vomit; my reason for this was to enable vomit to overlap with exit, which could block flames. Hypothetically, if the spec were changed so that Penelope could say climb up a wall and overlap with it, there could be a bug.

//I’m also assuming level finished and player died returns will automatically call cleanup for me

4.

For abstract base types, I added new Actor, Zombie, Goodie, and Projectile to see if there was a compilation error, and there was.

StudentWorld:

I tested to see if I could load a level. I then tested to see if I could move objects in the game. I also tested to see if the graphics would work as planned. I also tested for blocking of movement by utilizing intersection of rectangular bounds as well as overlap of objects. I also tested to see if all movable actors could move properly. I didn’t delete dead actors immediately, but set their statuses to dead, which would tell doSomething() to return immediately, and have move() delete all dead actors and erase the slot from list.

Note: most of description below refers to doSomething(), which overwrote the pure virtual functions.

Penelope: I tested if player responded to arrow keys, if Penelope could collide into zombie/citizen. I also tested whether she picked up a goodie or could exit. I also tested her flamethrower to see if flames spewed. I also tested to see if she deployed a mine when tab was hit. I then checked to see if vaccine count decreased whenever she hit enter and if her infected count restarted to 0 when she used one. I also got near a zombie and had it vomit on her to see if infected count would increment. I also checked if flames and mines would decrement after usage. I also checked to see if she died from flame from explosion and if she could die by overlapping with pit. I also checked to see if she made her death sound whenever she died and subsequently lost a life; I also tested to see if 0 lives meant game over.

Wall: I checked to see if any actors were blocked by moving into wall. I checked to see if wall blocked flames. I also checked to see if wall blocked flames from mine and vomit.

Exit: I checked to see if actors could overlap with exit. I checked to see if citizens could leave level and then increase score by 500. I checked to see if Penelope could exit when no citizens available and when she was unable to exit because some citizens needed to be saved. I also checked to see if it blocked flame but not vomit. I had exit use its own overlap function, different from the one Pit uses because exit overlap had very different functionality: it only takes in Penelope and citizens and increases score and causes citizen saved sound, whereas Pit kills actors regardless of type.

Citizen: I checked to see if citizens could die from flames, pit, landmine, and zombie vomit. When zombie vomited and got citizen infected for first time, I checked to see if citizen made got infected sound just once. I also checked to see if both dumb and smart zombies were born whenever it died from infection. I also checked to see if it ran to me when Penelope was close and ran away if closest zombie was closer than Penelope. I also checked to see if death caused me to lose points.

Flame: I tested to see if flames could be created on exits and walls when I hit space. I also tested when landmine exploded to see if up to 9 flames were created. I tested to see if it killed any goodie, zombie or citizen when I hit flamethrower or let them overlap with mine. I tested to see if flame would last 2 ticks and if sound was made from flamethrower

Vomit: I tested to see if vomit was created and infected Penelope by checking her infect count. I also checked to see if vomit created the sound vomit. I also checked to see if citizens infected by vomit turned into zombies

Dumb/Smart Zombie: I checked to see if zombies called vomit when human was nearby in its direction. I checked to see if its movement plan changed and if it vomited when human was nearby. I also checked to see if moveplan was reset because it ran into a wall or other blocking actor. For dumb zombie, I checked to see if it moved randomly, while for smart zombie, I checked to see if it pursued human if within its 80-pixel distance. I also checked to see if score increased by 2000 for smart and 1000 for dumb, and I checked if its death led to new vaccine in random direction.

All Goodies: I checked to see if they could be overlapped by other actors besides wall of course. I checked to see if only Penelope could pick it up and if score increased by 50. For landmine, I checked to see if my landmines increased by 2, vaccine increased by 1, and gas increased flame by 5. I also checked to see if vaccine was wasted and if it properly reset infect Count and decremented from usage. I also checked to see if hitting space called up to three flames, provided it wasn’t blocked by wall or exit, and if landmine were deployed, I would have 30 ticks before next overlap by actor caused explosion

Landmine: I checked to see if overlap from human or zombie caused explosion sound, if up to 9 flames were created if no blockage, and if a pit were made right after. I also checked to see if flame caused immediate explosion, whether or not mine is active, such as firing flame right after setting mine or waiting for a minute and then firing a flame.

Pit: I checked to see if any human or zombie died by overlapping with it. If it did, then their statuses would change. I also checked to see if it did not block flame/vomit. I had pit use its own overlap function to reset actors’ statuses and cause proper sound and score changes