Eric Yang

304263623

1. The Actor class:
2. doSomething() is a pure virtual function because every type of actor has a different implementation of this function.
3. isAlive() is used to check if the actor is still alive. Each type of actor uses the same function.
4. setDead() is used to let the game know that the actor has died. Each type of actor uses the same function.
5. getWorld() allows each actor to access the StudentWorld object they are contained in. Each type of actor uses the same function.
6. revealExit() is the function that changes the exit from invisible to visible. This virtual function doesn’t do anything for any actor expect the Exit class.
7. isVisible() checks if the actor is visible in the game. This virtual function doesn’t do anything for any actor expect the Exit class.
8. applyBugSpray() takes action when the actor is affected by bugspray. Each actor implements it differently so this is virtual.
9. blocksPlayer() checks if the current actor is capable of blocking the player from moving onto the spot. This is virtual because it returns false but some actors have different implementations.
10. colocationKillsPlayer() checks if the current actor kills the player if they are in the same spot. This is virtual because it returns false but each actor has a different implementation.
11. blocksZumiandSprayers() checks if the current actor blocks a zumi from moving to the same location or a sprayer from being dropped in the same location. This is virtual because it returns false but some actors have a different return value for this function.
12. allowsSprayToBeDropped() checks if bug spray can be dropped onto the actor. This is virtual because it returns true but some actors return a different value.
13. allowsSprayToPass() checks if bug spray can be sprayed beyond the current actor. This is virtual because all actors return true expect bricks.

The Player class:

1. doSomething() takes the player’s turn for the current tick. It checks for input and if the player has died.
2. applyBugSpray() sets the player as dead.
3. activateWalkThroughWalls() sets the wall lifetime counter so that the player can walk through destroyable walls for that set amount of ticks.
4. canWalkThroughWalls() checks if the player is currently able to walk through walls.
5. IncreaseSimultaneousSprayers() increases the number of sprays a player can place for a set amount of ticks.
6. dropSprayer() creates a sprayer at the current location if the number of sprayers hasn’t exceeded the maximum.
7. addSprayer() gives the player another sprayer once a sprayer detonates on the map.

The Brick class:

1. doSomething() does nothing for bricks
2. colocationKillsPlayer() returns true. This function is virtual because a player can stand on destroyable bricks if he has the ability to walk through walls.
3. blocksZumiandSprayers() returns true since bricks block zumis and sprayers.
4. allowsSprayToPass() only returns false for bricks since spray can’t go past them.

The PermanentBrick class:

1. blocksPlayer() always returns true since a player can’t stand on a permanent brick.
2. allowsSprayToBeDropped() always returns false since bug spray can’t affect permanent bricks.

The DestroyableBrick:

1. applyBugSpray() sets the brick to dead.
2. blocksPlayer() returns true if the player does not have the walk through wall ability and false is the player does.
3. colocationKillsPlayer() will return true when the player is standing on the brick and his ability runs out

The Exit class:

1. doSomething() will set the level as finished if the exit and visible and the player is standing in the same location
2. isVisible() returns true if the exit is visible in the game
3. revealExit() sets the exit to visible

The TimedLifetimeActor class:

1. doSomething() does nothing since sub classes all have different implementations of this function.
2. expireImmediately() sets the lifetime to 0.
3. decreaseLifetime() decreases the lifetime by 1.
4. getLifetime() returns the current lifetime.

The BugSprayer class:

1. doSomething() will decrease the lifetime by 1 and check if the lifetime is 0. If it is 0, the Sprayer creates a spray in is position and sprays in the adjacent areas 2 squares over. Another function checks if the sprays can be dropped and if they can travel 2 squares over. Then the function gives the player a sprayer and sets the sprayer to dead.
2. applyBugSpray() causes the sprayer to immediately die.
3. blocksZumiAndSprayers() always returns true for sprayer objects since zumi and other sprayers can’t be stacked on top of a sprayer.

The BugSpray class:

1. doSomething() decreases the lifetime by 1 and sets the spray to dead if the lifetime is 0. Otherwise, the function applies spray to the current spot.

The ExtraLifeGoodie class:

1. doSomething() decreases the lifetime by 1 and sets the goodie to dead if the lifetime is 0. Otherwise, it checks if the player is on the goodie. If the player is, the player gains one life, the goodie expires, and increases the player’s score.

The WalkThroughWallsGoodie class:

1. doSomething() decreases the lifetime by 1 and sets the goodie to dead if the lifetime is 0. Otherwise, it checks if the player is on the goodie. If the player is, the player gains the ability to walk through destroyable bricks for a set amount of ticks, the goodie expires and increases the player’s score.

The IncreaseSimultaneousSpraysGoodie class

1. doSomething() decreases the lifetime by 1 and sets the goodie to dead if the lifetime is 0. Otherwise, it checks if the player is on the goodie. If the player is, the player gains the ability to place more sprayers by a certain amount for a set amount of ticks, the goodie expires and increases the player’s score.

The Zumi class:

1. applyBugSpray() sets any zumi to dead.
2. blocksPlayer() returns false since zumis and the player can be in the same spot
3. colocationKillsPlayer() returns true since a zumi kills the player.
4. randInt() selects a random integer between the two inputted ingeters inclusive
5. increaseTick() keeps track of the Zumi’s turn and returns true if it is time for the Zumi to do something

The SimpleZumi class:

1. doSomething() checks if the zumi is currently at the same spot as the player and kills the player if it’s true. The Zumi moves if the increaseTick() function returns true. The Zumi tries to move in the direction it is currently set to. If it can’t move in the direction, randomly select a new direction.

The ComplexZumi class:

1. doSomething() does the same as the one for SimpleZumi because it has not been implemented.
2. The ComplexZumi class has not been implemented yet, so it behaves like a SimpleZumi.
3. I didn’t know how to move on to the next level, so I decided to create my own function that converts integers to strings. I used this function to create the level\_\_.dat string to load the next level.