Final Design Rationale Document by Owen

**Shotgun**

Files to be created:

* Shotgun( extends weapon item)
  + To be an item carried around
* ShotgunDirectionAction (extends action)
  + To find out all the location to shoot
* ShotgunHitAction (extends Action)
  + To find the actors in the location and hurt them
* ShotgunMenuAction(extends Action)
  + To make a mini menu for the shotguns

Other files involved:

* Action
* Map (passed)
* Actor (passed)
* Location
* GameMap (passed)

How should it Work:

Creating a shotgun as a weapon item will make the shotgun being able to be save in the inventory and can be carried around. When having that item, there should be more dropdown menu for shotguns and asking where to hit. After triggering the hit direction and getting all the location of where to shoot, then the execute should call the shotgunHitAction where we should know that is there any actor in the location received. Then we shoot them if they were there.

Design Ideas/Design Rationale

*Single Purpose Principle*

Making this have 2 actions will make each action class only one specific thing to do. Although that this project can be easier if doing everything (searching for location, find actor and damage) but it wouldn’t fulfill this principle so that I think it is better if we make separate file each and also I don’t want to overcomplicate stuffs so I instead decide to join the actor finding and hurt because for me, it falls under the same category and should be in a single action file.

*Open/Closed Principle*

Making this in this way will help not to break our own code. It will help because it is hard to change the signatures of each code. This again will also make it easier to be added more functionalities which covers the Open principle. There are a lot of open ended actions in the game and I think these too are actions that can upgraded or further implemented in the future whilst keeping its’ signature line safe.

*Don’t Repeat Your Self Principle*

In this task I will try to not need to create new classes that don’t extend anything. I think we can really rely on the engine sometimes to do things we want for some certain instances so that we should not redo the things that have been done but just using them instead. It is unnecessary to redo thing that have been done before and shouldn’t be doing that. I also don’t think that the actions I’m making have ever been done before.

**Sniper**

Files to be created:

* Sniper ( extends weapon item)
  + To be an item carried around
* SniperFindTargetAction (extends action)
  + To find out all the undead actor
* SniperAimAction (extends Action)
  + To aim at a certain actor
* SniperMenuAction(extends Action)
  + To make a mini menu for the sniper
* SniperHurtAction(extends Action)
  + To hurt the target selected

How should it Work:

Creating a sniper as a weapon item will make the shotgun being able to be save in the inventory and can be carried around. When having that item, we need to find all the actor with UNDEAD capability and save. Next there should be more dropdown menu for sniper to aim at each of the enemy and show a shoot if there have been an aim before. If they choose to aim, then we call the sniper aim action. If they choose Shoot, call sniper hurt action.

Design Ideas/Design Rationale

*Single Purpose Principle*

Making 4 actions on executing the sniper, excluding the one for menu, makes each action a single role to fulfill and no repetition use. We can do it all under one action file but then it will break this principle so I decided to make this like this.

*Encapsulation Boundaries*

If there are no restriction on cannot edit, I would like to add few stuff to the engine to make finding all actor on location instead of needing to search each location. But knowing that there are ‘boundaries’ of what we can do, I decided and maybe forced to make it the hard way but in regards, using the principle of encapsulation

*Open/Closed Principle*

The idea is so that in the future, we can develop the sniper into greater heights and this code can be part of it. That is the part of Open principle. Next is when we develop the next parts of the code, we have to be sure that the initial code doesn’t break down, therefore I need to make sure that this will not break.