FIT2099 A3 DOTPOINT PLAN – Paul M

USED ICONS

* . dirt
* # fence
* + tree
* \* crop
* o food
* F farmer
* Z zombie
* @ player
* H human
* ) plank
* - zombie arm
* | zombie leg
* , concrete
* X wall
* = door
* C car
* S shotgun
* s shotgun ammo
* R sniper rifle
* r rifle ammo

shotgun spreads

AXXYZZB  
 AXYZB  
 AYB  
 O

XYYB  
ZXYB  
ZZXB  
AAAO

CITY MAP

* Map components
  + Will need to make new Ground types to represent city terrain.
    - Existing ground types are Dirt, which is a floor and allows players to walk on it, tree which has a growth property, and Fence, which overrides canEnterActor to return false, making it impassible.
    - New types
      * Concrete – will be a floor type functionally identical to dirt, with a different icon ‘,’
      * Wall – will be functionally identical to fence, with a different icon ‘X’
      * Door – same as floor =
      * Car - C
    - These components will be added to the existing instantiation of groundFactory, in Application
  + Map will be named gameMap2, will be an instantiation of GameMap in the same manner as the existing gameMap in Application
  + Will make it the same size as the existing map, for sake of consistency
* Moving between maps
* REVISION – I think I want to make this a MoveActorAction instead. Less hacky, better for duplication if more cars need implementation. Car class and ability to call said action should be tied together.
* REWORKED – it’s an item now. Benefits of this include no dependency on Groundfactory, no chance for the exit location and car sprite to mismatch, and better DRY adherence.
  + ~~Car will be a ground type at a location on the map.~~
    - ~~This is because the car itself does not move, and Exit class will be used to travel between maps, and searching for valid Exits in processActorTurn in World searches the current Location for exits~~
      * ~~Therefore basing map travel Exits on location will not need a separate new implementation of Exits, fulfilling DRY.~~
  + ~~At world creation, location of car on a given map will use Location.addExit to add a new exit to the map.~~
    - ~~Location of the car on the other map will be the destination for that exit.~~
  + Shotgun and Rifle will be placed on the map in separate buildings, ammo sprinkled around, a few zombies sprinkled around. Humans may also be sprinkled as they won’t pick up firearms.

GUNS

* Make an abstract class, Firearm extends Item.
  + This will hold the mechanics for shooting and checking ammo. The precise type of shot, unique to each of the two weapons, will be covered in the extension of the abstract class into its subclasses. This fulfills DRY and RED, by removing the need to rewrite menu commands for each firearm, and ensuring that future implementation of firearms will always be dependent on a consistent, unchanging abstraction.
  + To fire a gun
    - New class, ReadyWeapon action. A check for a Firearm will need to be made in world.ProcessActorTurn(). If there is a firearm in player inventory, and ammunition for that corresponding firearm, one ReadyWeapon action per firearm can be added to actions list passed to actor.playturn.
    - This will be dependent on a check for if the actor is human, as zombies can’t shoot guns.
    - ReadyWeapon action execution will call the shoot method in the specific firearm subclass. The exact results of this will depend on the firearm in question, but it will be a new menu with a variety of options
      * Direction of fire for shotgun, choice of targets then choice of aim or fire for rifle
  + Being a firearm (not a weapon) prevents Zombies from picking them up.
* Shotgun
  + 75 damage seems reasonable, it’s a big chunk but not a 1hko as specified.
  + Shoot method opens a menu showing the possible directions to shoot the gun, 1 2 3 4 6 7 8 9 using same directional keybinds as walking.
  + On selecting fire direction, Location in that immediate direction plus Location tiles in the associated pattern will check for presence of an actor, then apply damage to that actor at a 75% chance based on random roll.
    - Spread of damage to next tile dependent on canEnterActor to ensure shots don’t go through walls, specific to which spreadline it is following as shown in diagram at doc top.
      * Eg in north shot, if X at range 2 is blocked but not A at range 2, X at range 3 will not be included in damage but A at range 3 will.
    - Buck spread as depicted at top of doc, as it can’t be easily depicted in dotpoints or this font.
* Rifle
  + 40/80/999 damage depending on aim level
  + Shoot method opens a menu showing possible targets, all zombies on the map, if a target is not already declared
    - Selecting a target stores the target as a variable in the rifle
      * Target stored in the rifle to reduce dependency
    - Option is then presented to fire, or aim
      * aimLevel is stored in the rifle, again to reduce dependency
      * if aimLevel = 1, the UI notes the current target, and offers the option to either fire or aim further
      * if aimLevel = 2 only the option to fire is presented
  + currentlyAiming will be a Boolean toggle added to Player to preserve aiming status.
    - playTurn in player can be modified. It no longer simply returns menu choice, but stores it and checks currentlyAiming before returning the stored menu choice. If the menu choice is not to aim further or fire, currentlyAiming can be flipped and a method called clearTarget() called in the rifle, which does what it sounds like.
      * This way, the Player and the Rifle only interact when something actually needs to be done to either. The rifle is not changing values in the player for every turn of aiming, and the player is not checking their rifle every time they move regardless of circumstance. Fulfills RED.
  + No plans to implement terrain collision as yet, as that’s more difficult than for the shotgun. Might add it if I get time.

ADDITIONS

Classes are first level dotpoint. Methods are second level dotpoint. Variables are third level dotpoint.

Italicized items are new creations, and so all methods and variables may also be considered new, and so for convenience are not listed. square brackets are comments for understanding

* *Concrete*
* *Wall*
* *Door*
* *Car*
* Application
  + - *map2*
    - groundFactory [added new groundtype parameters]