Question 1. FPS Game

model Play

abstract class FPSState

operations

seeOpponent(defeatable:Boolean)

begin

end

lostOpponent()

begin

end

receiveBlow(health:Integer)

begin

end

end

class neutral < FPSState

operations

seeOpponent(defeatable:Boolean)

begin

if (defeatable) then

WriteLine('See opponent, and is defeatable. Attack!');

self.play.fpsState := self.play.attackState

end;

if (defeatable = false) then

WriteLine ('See opponent, and is not defeatable. Panic!');

self.play.fpsState := self.play.panicState

end;

end

lostOpponent()

begin

WriteLine('In state:neutral operation:lostOpponent There is no opponent. You must find one first.')

end

receiveBlow(health:Integer)

begin

WriteLine('In state:neutral operation:receiveBlow There is no opponent. You must find one first to receieve a blow.')

end

end

class attack < FPSState

operations

seeOpponent(defeatable:Boolean)

begin

end

lostOpponent()

begin

WriteLine('Lost to opponent, but not dead. Return to Idle');

self.play.fpsState := self.play.neutralState

end

receiveBlow(health:Integer)

begin

if (health = 0) then

WriteLine('Received blow from opponent, health at 0 die.');

self.play.fpsState := self.play.dieState

end;

end

end

class panic < FPSState

operations

seeOpponent(defeatable:Boolean)

begin

end

lostOpponent()

begin

WriteLine('Lost to opponent, and not dead. Return to Idle.');

self.play.fpsState := self.play.neutralState

end

receiveBlow(health:Integer)

begin

if (health >= 0) then

WriteLine('Received blow, and in a panic. Dead!');

self.play.fpsState := self.play.dieState

end;

end

end

class die < FPSState

operations

seeOpponent(defeatable:Boolean)

begin

end

lostOpponent()

begin

end

receiveBlow(health:Integer)

begin

end

end

class Objective

attributes

operations

end

class Save

attributes

operations

end

class LoadOut

attributes

operations

end

class Level

attributes

operations

end

class Enemy

attributes

operations

end

class DestructableEnvironment

attributes

operations

end

class Stockpiles

attributes

operations

end

class Play

attributes

health : Integer init : 10

fpsState : FPSState

neutralState : FPSState

attackState : FPSState

panicState : FPSState

dieState : FPSState

operations

initInstance()

begin

self.neutralState := new neutral;

self.attackState := new attack;

self.panicState := new panic;

self.dieState := new die;

self.fpsState := self.neutralState; -- We are starting in neutral

end

seeOpponent(defeatable:Boolean)

begin

self.fpsState.seeOpponent(defeatable);

end

lostOpponent()

begin

self.fpsState.lostOpponent();

end

receiveBlow(health:Integer)

begin

self.fpsState.receiveBlow(health);

end

statemachines

psm StatePattern

states

startUp:initial

neutral [self.fpsState = neutralState]

attack [self.fpsState = attackState]

panic [self.fpsState = panicState]

die [self.fpsState = dieState]

transitions

startUp -> neutral {create}

neutral -> attack { [health > 0 and defeatable = true] seeOpponent()}

neutral -> panic { [health > 0 and defeatable = false] seeOpponent()}

neutral -> neutral { [fpsState = neutralState] lostOpponent()}

neutral -> neutral { [fpsState = neutralState] receiveBlow()}

attack -> neutral{ [health > 0 ] receiveBlow()}

attack -> die { [health <= 0] receiveBlow()}

attack -> attack { [fpsState = attackState] seeOpponent()}

panic -> neutral{ [health > 0 ] receiveBlow()}

panic -> die { [health <= 0] receiveBlow()}

panic -> panic { [fpsState = panicState] seeOpponent()}

die -> die { [fpsState = dieState] seeOpponent()}

die -> die { [fpsState = dieState] receiveBlow()}

die -> die { [fpsState = dieState] lostOpponent()}

end

end

-- associations

association fpsState between

FPSState[1]

Play[1]

end

association choiceLoad between

Play[1] role playSpec

LoadOut[\*] role gear

end

association saveChoice between

Play[1] role playSpec

Save[\*] role save

end

association objectives between

Play[1] role playSpec

Objective[\*] role goalObject

end

association level between

Play[1] role playSpec

Level[1] role levelOn

end

association enemies between

Play[1] role playSpec

Enemy[\*] role enemy

end

association destructableObjects between

Play[1] role playSpec

DestructableEnvironment[\*] role objects

end

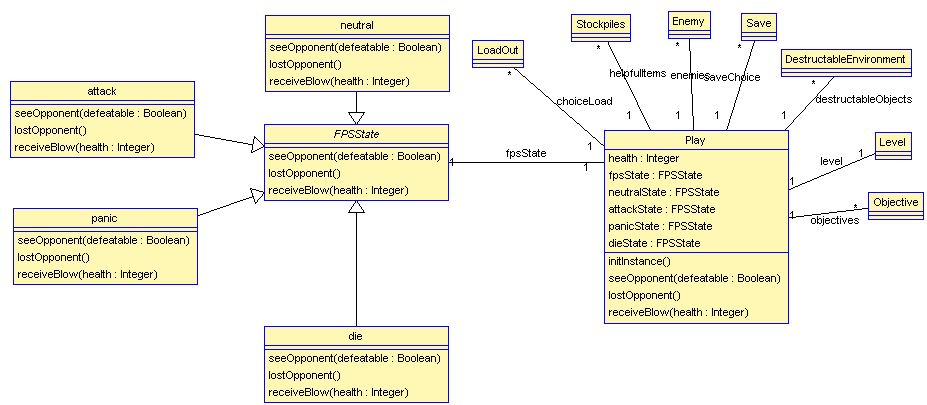
association helpfulItems between

Play[1] role playSpec

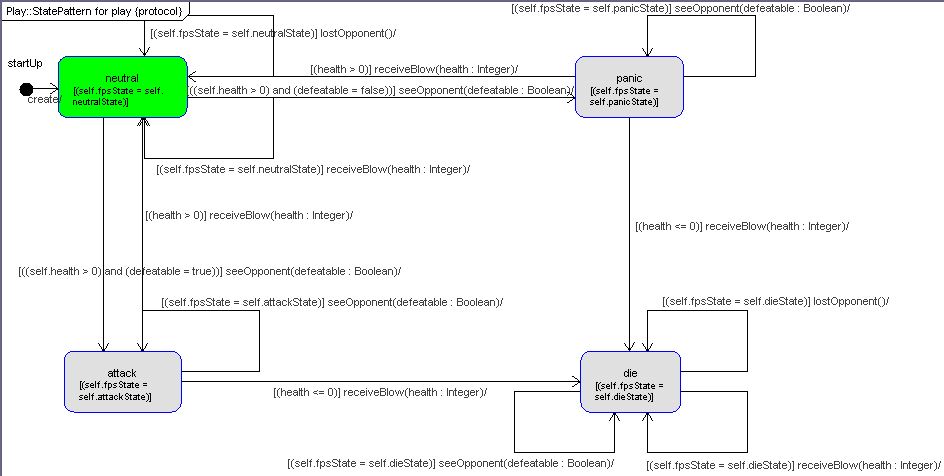
Stockpiles[\*] role pickUps

End

Class diagram



State Machine



Question 2.