System Specification Document

Domain Model Overview

Player

Road

Damages

Follows

Gets Reward

Enemy

Attacks

Tower

Knows

Knows cost and type

Round

Upgrades

Shop

CashVault

Holds

Delgates

Adds Enemies

Takes when enemy dies

Game

Notfy movement and death

TowerNotifier
(Mediator)

Subscribe Tower

Board

Add Tower

RoadToken

section: Road
isFinsihed: boolean
distance int

advance()
getPosition()

<<Road>>
Interface

advance(RoadToken token, int distance)
getPosition(int distance)

Uses

BaseEnemy

RoadSpawn

RoadSection

RoadEnd

<<Enemy>>
Interface

isFinsihed()
move()
attack()
getPosition()
isAlive()

Uses

RoadBuilder

board: Board,
player: Player,
points: Points

RoadBuilder(board: Board, player: Player),
Add(Point point)
Build(): RoadSpawn

InvalidRoadException

Player

int health

damage()
isAlive()

Road

Road consists of a generalized linked list of road sections which is converted from an arbitrary length of 2d points by the builder. A RoadToken moves through the linked list oblivious of change of sections and position and only advances by wanted distance. It is notified when it arrives at the roadend.

Enemy

Handles the health, position dynamic through use of RoadToken, and reward of enemies. Interface is the main api towards other components.

Round

Road consists of a list of Round objects which in turn consists of RoundEvents which specify time and hold an Enemy Factory.

Cashvault

At the moment, the cash vault is given a list of dead enemies removed from the game loop. An improvement would be to make the cash vault an observer of enemy health instead.

Game

This is where the game loop subscribes to newly spawned enemies from the Round. It is an abstraction that handles the update of movement of alive enemies through delegation.

Board

Controls update of towers, and ensure valid position of them.

Shop

Mediates the interaction towerfactory, cashvault, and board. Propagates exception form cashvault and towerfactory for invalid placement or overdraft.

Tower Notifier

Mediates the subscription and unsubscription of towers to alive enemies to prevent dependency cycle between Board and Game. The reason for having two separate update handlers for enemy and towers is that we wanted to have towers that receive other input than enemies, e.g. boost

nearby towers, and thus there need to be an explicit method call from the game loop on towers rather than just doing observer notifications.

Tower

Delegates attack modes through composition, and upgrades through a state pattern. Interface is the main api towards other components.