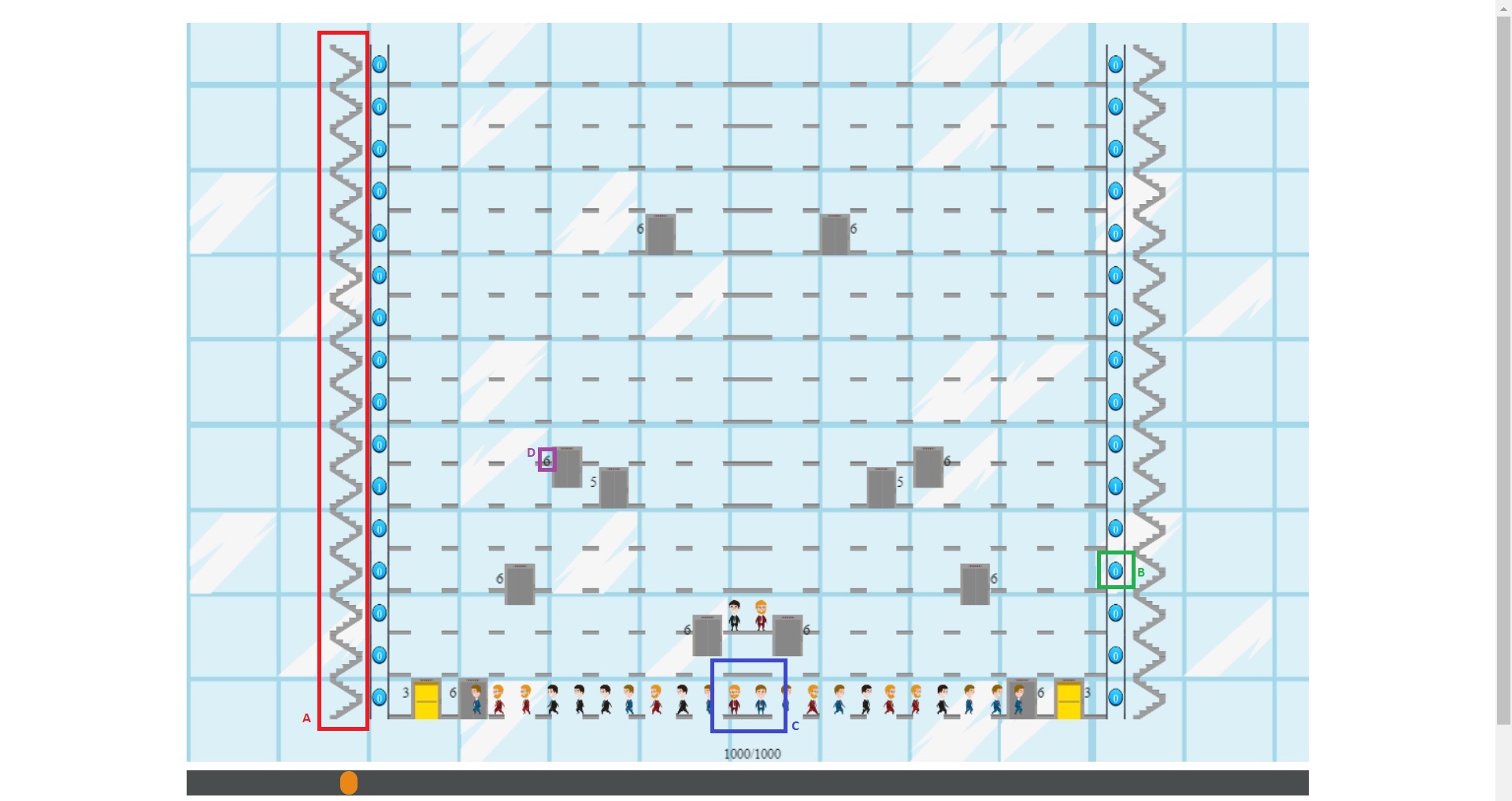
# **SBERBANK HACKATHON 2018**

Номинация «Лучший разработчик»

# 

# Модель мира

В здании есть 14 лифтов (по семь с каждой стороны), 16 этажей, две точки появления пассажиров на каждом этаже и две лестницы. Здание симметрично относительно центра. Слева находятся лифты, которыми управляет первый игрок, справа -- лифты второго игрока. Точки появления пассажиров также принадлежат либо первому игроку (слева), либо второму (справа). Управлять можно только своими лифтами, лифт поддерживает две команды: забрать определенного пассажира ("своего" или "чужого") или уехать на определенный этаж (с пассажирами или без). Напрямую пассажирами управлять нельзя. Задача -- заработать как можно больше очков за перевоз пассажиров.



А – лестничная клетка

B – счетчик сотрудников на этаже

C – зона появления новых пассажиров

D – счетчик пассажиров в лифте

# Лифт

## Параметры лифта

id (int)

Уникальный среди всех лифтов идентификатор.

У левого игрока: 1, 3, 5, 7, 9, 11, 13

У правого игрока: 2, 4, 6, 8, 10, 12, 14

type ("FIRST\_PLAYER" либо "SECOND\_PLAYER")

Принадлежность игроку

state (int) - Этот параметр определяет текущее состояние лифта:

0, waiting

В этом состоянии лифт находится краткое время (1 тик в данный момент) между закрытием дверей и отправкой на заказанный этаж

1, moving

В этом состоянии лифт находится, когда он едет на этаж

2, opening

В этом состоянии лифт находится, когда он открывает двери

3, filling

В этом состоянии лифт находится, когда он открыл двери и готов к заходу пассажиров внутрь

4, closing

В этом состоянии лифт находится, когда он закрывает двери

Таким образом, лифт начинает свою игру с состояния 3, и в дальнейшем переходит в состояния 4 -> 0 -> 1 -> 2 -> 3 Позвать пассажира в лифт можно лишь в состоянии 3 Назначить лифту следующий этаж можно в состоянии 3, при этом он начнет закрывать двери и перейдет в состояние 4

floor (int)

Текущий этаж, на котором стоит или который проезжает лифт Этот параметр меняется при движении лифта вверх или вниз, как только лифт пересекает границу соответствующего этажа (встает в положение, в котором он стоял бы на этом этаже, если бы остановился)

y (float)

Параметр y определяет положение лифта по вертикальной оси. Именно этот параметр плавно изменяется при движении лифта с этажа на этаж. “y” является дробным и измеряется в том же масштабе, что и этажи. То есть, при движении лифта с 1 этажа на 3 параметр y будет дробно менять своё значение каждый тик, при этом параметр floor изменится ровно 2 раза - 1->2 и 2->3

speed (int)

Скорость движения лифта, измеряется в величине y, которую лифт пройдет за 1 тик. Скорость лифта зависит от количества и веса пассажиров, формулу можно найти ниже в документации

next\_floor (int)

Следующий требуемый этаж. Стоит обратить внимание, что следующий требуемый этаж меняется пользователем, а не системой. То есть, при необходимости отправить на 7 этаж лифт, который в данный момент стоит на 3, в процессе движения лифта будут меняться переменные y и floor, но не next\_floor

time\_on\_floor (int)

Количество тиков, которые лифт провёл на текущем этаже. Отсчет начинается в самом начале игры и каждый раз, когда лифт выходит из состояния 1

passengers (list)

Массив со всеми пассажирами, находящимися в лифте в данный момент. Состав массива изменяется, когда пассажир заходит в лифт или выходит из него

## Методы лифта

go\_to\_floor

Данный метод приказывает лифту двигаться на указанный игроком этаж. Приказать двигаться можно лишь своему лифту. Менять направление движения или делать остановки, когда лифт уже находится в состоянии 1, в данный момент нельзя

# Пассажир

## Параметры пассажира

id (int)

Уникальный среди всех пассажиров идентификатор

type ("FIRST\_PLAYER" либо "SECOND\_PLAYER")

Принадлежность игроку

state (int)

Этот параметр определяет текущее состояние пассажира:

1, waiting\_for\_elevator

В этом состоянии пассажир ждет лифта. Новые пассажиры появляются на этаже именно в этом состоянии

2, moving\_to\_elevator

В это состояние пассажир переходит, когда игрок отдает команду лифту "пригласить" пассажира методом set\_elevator. Если пассажир в этот момент находится в состояниях 1 или 3, он выбирает ближайший лифт из всех предложенных на этом тике и начинает идти к лифту, переходя в состояние 2

3, returning

В это состояние пассажир переходит, если "пригласивший" его лифт уехал без него. Тогда пассажир в состоянии 2 идет обратно к своему месту ожидания на этаже, переходя в состояние 3. В этот момент его может пригласить другой лифт

4, moving\_to\_floor

Пассажир идёт по лестнице. Если с момента появления пассажира на этаже прошло 500 тиков и он все еще не находится в лифте, он перестаёт ждать лифта и идёт по лестнице. Перемещение пассажира с этажа на лестницу при этом происходит мгновенно. Перемещение с лестницы на этаж также мгновенное.

5, using\_elevator

Пассажир находится в лифте. Если пассажира "пригласил" один из лифтов и пассажир сумел в него войти (координаты x пассажира и лифта сравнялись, когда лифт был в состоянии 3), то пассажир считается зашедшим в лифт и будет ждать, когда лифт поедет на этаж. Если лифт остановится на том этаже, на который пассажиру нужно, то пассажир после открытия дверей перейдет в состояние 6, потратит какое-то количество тиков на выход из лифта и скроется на своём этаже. Там он проведёт 500 тиков, после чего захочет на другой этаж и появится снова в состоянии 1 на том же этаже, на котором он вышел из лифта

6, exiting

Пассажир выходит из лифта. Для этого ему нужно некоторое количество тиков (40 в данный момент)

weight (float)

Вес пассажира (по факту - коэффициент замедления). См.раздел про подсчет веса.

elevator

id лифта, к которому в данный момент идёт пассажир, или в котором он уже находится. Лифт назначается в момент вызова метода set\_elevator, если соблюдены все необходимые условия

floor (int)

Этаж. Аналогично параметру floor у лифта

y (float)

Аналогично параметру y у лифта

x (int)

Положение по горизонтальной оси. 0 находится в центре этажа. Скорость изменения этого параметра (передвижения пассажира) - две единицы в тик

time\_to\_away (int)

Время, отведенное пассажиру на ожидание лифта. Изначально этот счетчик равен 500 и уменьшается каждый тик, который пассажир проводит в состояниях 1, 2 или 3. Если счетчик становится равен нулю и пассажир при этом в состояниях 1, 2 или 3, пассажир уходит на лестницу

from\_floor (int)

Последний из этажей, на котором пассажир начал ждать лифта

dest\_floor (int)

Этаж, на который пассажир хотел бы отправиться

## Методы пассажира

set\_elevator(elevator)

Метод, использующийся для приглашения пассажира в лифт. Если все условия соблюдены, пассажир начинает идти к лифту. Принимает на вход объект лифта

has\_elevator

Метод, использующийся для проверки, назначен ли лифт пассажиру. Возвращает true или false

# Механики игры

## Вес и скорость лифта

Каждый пассажир обладает определенным весом, который обязательно больше единицы. Этот вес можно считать коэффициентом уменьшения скорости лифта, везущего пассажиров вверх. Вес каждого пассажира равен случайной величине от 1.01 до 1.03 включительно. При вычислении скорости лифта, количество тиков, необходимое, чтобы проехать один этаж, умножается последовательно на вес каждого из пассажиров. Например, если один пассажир весит 1.03, а второй 1.02, то скорость лифта следует умножить на вес обоих.

Проще всего это показать на количестве тиков, за которые лифт пройдет 1 этаж: timeForFloor = 50 \* 1.03 \* 1.02 = 52.53 тиков на этаж вместо изначальных 50.

Если лифт везет более 10 пассажиров, то это время нужно умножить еще на 1.1.

Счетчик тиков до следующего этажа округляется до целого вверх (для всех этажей, не для каждого). Таким образом, скорость 52.53 означает, что лифт доедет до этажа через 53 тика, а два этажа проедет за 52.53 \* 2 = 105.06 - 106 тиков.

Обратите внимание, вниз лифт едет всегда с одной и той же скоростью, равной скорости движения лифта без пассажиров.

## Система наценки очков

За каждого перевезенного на нужный ему этаж пассажира начисляется определенное количество очков - floors \* 10. Здесь floors это abs(dest\_floor-from\_floor), то есть при перевозке с 2 этажа на 7 игрок получит 5\*10=50 очков.

При перевозке чужого пассажира очки за его перевозку удваиваются.

! Обратите внимание, других способов заработать очки во время игры, нет.

## Свои и чужие пассажиры

Каждый пассажир может появиться на этаже чуть левее или чуть правее центра (появление чуть левее означает, что это пассажир левого игрока, а чуть правее - что это пассажир правого игрока). Чужого пассажира увезти сложнее, так как для этого нужно простоять на этаже минимум 40 тиков с открытыми дверьми (в состоянии 3). Кроме того, объективно свои пассажиры находятся чуть ближе к своим лифтам.

При этом за чужого пассажира даётся в два раза больше очков при перевозке

Обратите внимание, после успешной перевозки на нужный пассажиру этаж, он становится вашим и в следующий раз появится на этом этаже именно с вашей стороны. Впрочем, противник может переманить его обратно к себе, выполнив те же условия

## Жизненный цикл пассажира

В начале игры пассажиры появляются парами в двух местах ожидания на первом этаже (по одному для каждого игрока), в статусе 1. Так происходит раз в 20 тиков вплоть до 2000 тика, таким образом в игре накапливается 200 пассажиров, по 100 для каждого игрока. Для каждого пассажира заранее определен упорядоченный список этажей, которые он хотел бы посетить в течение игры (от 1 до 5 случайных уникальных этажей).

* Распределение количества этажей и самих этажей равномерное.
* Этот список всегда заканчивается первым этажом. То есть, пассажиры появляются на первом этаже и хотели бы вернуться на первый этаж после всех своих путешествий.
* Обратите внимание, игра может закончиться и до того, как все пассажиры вернутся обратно на первый этаж.
* Как только пассажир появится на этаже, начинается отсчет длиной в 500 тиков. Если за это время пассажир не уедет на лифте, он уходит на нужный ему этаж по лестнице. Скорость движения пассажира по лестнице вверх равна 200 тиков на этаж, вниз - 100.
* После того, как пассажир оказался на нужном этаже (приехал на лифте или пришел по лестнице), он отправляется 500 тиков "гулять" по этажу. Потом он снова появляется на этаже, чтобы поехать еще куда-то.
* Если пассажир приехал на лифте, ему нужны дополнительные 40 тиков, чтобы выйти из лифта.

## Жизненный цикл лифта

В начале игры все лифта появляются на 1 этаже в состоянии 3. Для передвижения на другой этаж лифту нужно закрыть двери, отправиться, а приехав - открыть двери. На закрытие и открытие дверей лифт тратит 100 тиков. Кроме того, лифт не может провести на этаже после открытия дверей менее 40 тиков.

* Лифт не может везти более 20 пассажиров за раз.
* Лифт не может больше "приглашать" пассажиров, если в нем уже находится 20. Пассажиры, которые уже идут в лифт в этот момент, дойдут до него, развернутся и перейдут в состояние 3.

## Расположение лифтов

Точка входа ближайшего к центру лифта каждого игрока находится в 60 единицах от центра этажа (или в 40 единицах от места ожидания своих пассажиров). Точка входа каждого следующего лифта игрока располагается в 70 единицах от точки входа предыдущего.

Оба места ожидания пассажиров на каждом из этажей этаже - на 20 единиц левее и правее центра.

# Работа стратегии

## Точка входа в стратегию

Каждый тик у стратегии игрока вызывается функция on\_tick или аналогичная, в которую поступают четыре массива:

* Свои лифты игрока
* Чужие лифты
* Свои пассажиры игрока
* Чужие пассажиры

Вызывая соответствующие методы у пассажиров и своих лифтов, игрок обеспечивает верную и оптимальную работу стратегии.

# Локальная отладка

server является tcp-сервером, написанным на python2.7+tornado. При запуске он ждет соединения от двух tcp-клиентов со стратегией, после чего отсылаем им команду "начали!" и делает всё, что надо. В конце он отсылаем обоим команду "закончили" и завершается, сбрасывая в visualizer/game.js лог игры.

И сервер, и оба клиента при этом завершают свою работу.

## Подготовка

Для успешной работы сервера нужно поставить необходимые пакеты из world/requirements.txt. Проще всего сделать это с помощью команды pip install -r world/requirements.txt

## Запуск сервера

После этого можно сразу переходить в папку world и запускать python run.py. Сервер запустится и будет терпеливо ожидать коннекта от двух стратегий. Кроме того, в run.py можно параметром передать число, характеризующее random seed текущей игры. Например, если раз за разом запускать run.py 1209328, то раз за разом можно оттачивать стратегию на одинаковом наборе пассажиров под номером 1209328. Seed игры можно подсмотреть в визуализаторе, как при локальном запуске, так и при запуске игр на сайте.

## Запуск стратегий

Работающая стратегия состоит из обвязки и стратегии. Стратегия занимается логикой, обвязка же просто запускает стратегию как tcp-клиент.

Обвязку можно взять из списка доступных клиентов - берем нужный нам, правим базовую стратегию Strategy.

Всё, tcp-клиент стратегии готов, можно запускать!

Таким образом нужно запустить любые две стратегии, они автоматом подконнектятся к tcp-серверу мира и начнут играть. Через некоторое время игра закончится и все три программы (два клиента и один сервер) завершатся, и можно будет посмотреть визуализацию игры.

## Просмотр игры

Реализован просто - сервер в конце игры пишет все нужные данные в visualizer/game.js. Чтобы посмотреть игру, достаточно открыть в браузере visualizer/index.html

# Ограничения для стратегии

В отдельной игре побеждает игрок, набравший за игру большее кол-во очков.

Решению выдается:

* 512 RAM на стратегию
* 1 поток на стратегию
* 10 секунд максимум на ход
* На все 7200 тиков алгоритму дается 120 секунд

