Министерство образования Республики Беларусь

Учреждение образования

«БелорусскиЙ государственный университет

информатики и радиоэлектроники»

Факультет инновационного непрерывного образования

Кафедра проектирования информационно-компьютерных систем

Дисциплина «Основы программирования информационных систем»

**КОНТРОЛЬНАЯ РАБОТА №1**

**«Программирование вероятностной доски Гальтона»**

Выполнил: Старовойтов Е.А.

Группа № 694351 - 2 курс

№зачетки 6943015

Проверила: Пархоменко К.А.

Минск 2017

Содержание

1. Введение. Устройство доски Гальтона и ее математическое обоснование

2. Объектно-ориентированные технологии программирования и используемый язык

3. Листинг кода (без кода используемых библиотек)

4. Список используемых источников

**Введение.** **Устройство доски Гальтона и ее математическое обоснование**

При выполнении работ мы затронем такие раздел математики как Теория вероятностей, которая изучает случайные события, величины, их свойства и операции над ними.

В наше время теория вероятности – это база на которой основаны анализ и прогнозирование.

Доска́ Га́льтона (англ. Galton box, также распространены названия квинкункс, quincunx и bean machine) — устройство, изобретённое английским учёным Фрэнсисом Гальтоном (первый экземпляр изготовлен в 1873 году, затем устройство было описано Гальтоном в книге Natural inheritance, изданной в 1889 году) и предназначающееся для демонстрации центральной предельной теоремы.

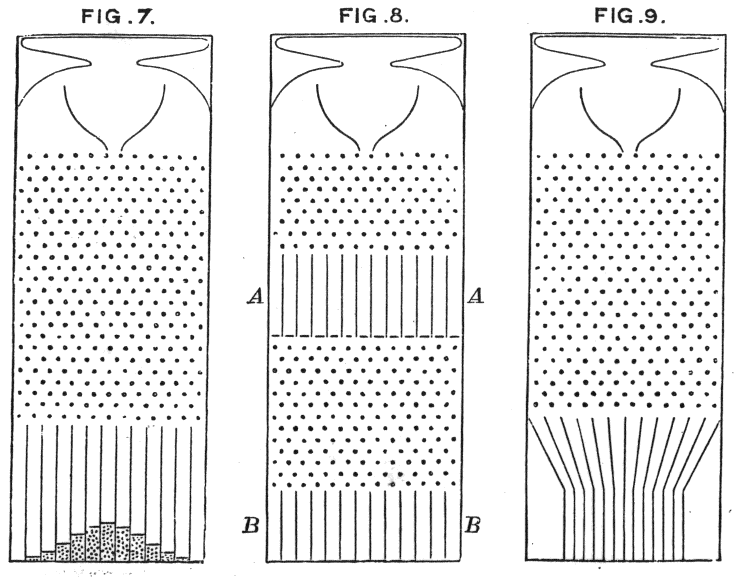
**Устройство**

Доска Гальтона представляет собой ящик с прозрачной передней стенкой. В заднюю стенку в шахматном порядке вбиты штырьки, образующие треугольник. Сверху в ящик через воронку (выход из которой расположен ровно посередине между левой и правой стенками) кидаются шарики. В идеальном случае сталкиваясь со штырьком, шарик каждый раз с одинаковой вероятностью может повернуть либо направо, либо налево. Нижняя часть ящика разделена перегородками (число которых равно числу штырьков в нижнем ряду), в результате чего шарики, скатываясь на дно ящика, образуют столбики, которые тем выше, чем ближе к середине доски (при достаточно большом числе шариков внешний вид столбиков приближается к кривой нормального распределения).

**Распределение шариков**

Обозначим как n общее число столкновений шарика со штырьками; как k число раз, когда шарик поворачивает направо (таким образом, он оказывается в k-м по порядку столбике). Тогда число способов, которыми он может добраться до k-го столбика определяется биномиальным коэффициентом . Отсюда следует, что вероятность оказаться в k-м столбике равна , где p — вероятность поворота направо (обычно можно считать, что  ). Это функция вероятностибиномиального распределения, которое в соответствии с центральной предельной теоремой при достаточно большом n аппроксимирует нормальное распределение.

Графическое представление устройства доски(взято с wiki):



**Объектно-ориентированные технологии программирования и используемый язык**

Обьектно-ориентированное программирование(ООП) – это такая методология программирования при которой программа представлена в виде совокупности обьектов, каждый из которых является экземпляром класса и которые между собой образуют иерархию наследования.

Объединение данных и свойственных им процедур обработки в одном объекте, называется инкапсуляцией и является одним из важнейших принципов ООП.

Основными фундаментальными понятиями характеризующие ООП являются:

**Класс** – это комплексный тип данных или шаблон на основе которых создаются конкретные программные обьекты. Он сотоит из набора полей (примитивных типов данных) и методов(функций для работы с этими полями). Каждый обьект имеющий структуру этого обьекты называется экземпляром класса.

**Наследование** – свойства класса которые позволяют создавать новые классы на основе уже существующих и дают возможность наследовать методы и свойства класса родителя.

**Инкапсуляция** – свойство обьектов системы, которое дает возможность обьединить данные и методы, работаюшие с ними в классе. Это позволяет использовать один метод для разных обьектов.

**Абстракция** – выделение значимых характеристик обьекта и абстрагирование от незначимых.

Полиморфизм - означает, что рожденные объекты обладают информацией о том, какие методы они должны использовать в зависимости от того, в каком месте цепоч-ки (дерева классов) они находятся, иными словами это концепция, реализующая "множество методов в одном интерфейсе".

**Модульность** – такая организация обьектов, которое заключает в себя полное определение их характеристик и методов внутри себя, это дает возможность сводного перемещения и повторного использования кода.

На данный момент, на многих языках программирования существует возможность следовать парадигме ООП. И хотя в некоторых случаях они отличаются по реализации, основные принципы остаются одинаковыми.

Самыми известными языкамипридерживающиеся ООП на данный момент являются C++ и Java, однако большую популярность на данный момент завоевывает Java Script, и хотя этот язык начинал свой путь лишь как инструмент для создание динамичности в Web, сейчас этот мощный инструмент который занимает второе место по популярности после Java.

В реализации работы были использованы дополнительные библиотеки такие как jquery и matter.js(физический движок для придания наглядности работы доски Гальтона)

**Листинг кода (без кода используемых библиотек)**

// обьявление используемых обьектов библиотеки Matter.js

var Engine = Matter.Engine,

World = Matter.World,

Bodies = Matter.Bodies,

Composite = Matter.Composite,

Composites = Matter.Composites,

Vector = Matter.Vector;

// переменные для создания характеристик шаров, границ и т.д.

var width = 750,

height = 650,

borderWidth = 20,

hexYOffset = 50,

hexXOffset = 350,

hexYSpacing = 38,

hexXSpacing = 44,

hexSize = 22,

boundaryHexSize = 27,

hexChamfer = 3,

ballPyramidRows = 18,

ballPyramidColumns = 24,

ballSize = 3.25,

ballFriction = 0,

ballBounciness = .25;

// обьявление стилей для используемых обьектов(цвет обьекта и цвет границ)

var boundaryStyle = {

fillStyle: '#F0F0F0',

strokeStyle: 'transparent'

};

var transparentStyle = {

fillStyle: '#777',

strokeStyle: 'transparent'

};

var ballStyle = {

fillStyle: '#6f71d1',

strokeStyle: 'transparent'

};

var wallStyle = {

fillStyle: 'transparent',

strokeStyle: 'transparent'

};

// создаем физический движок в заданной области(по найденному элементу)

var engine = Engine.create({

render: {

element: document.getElementById("player"),

options: {

height: height,

width: width,

background: 'transparent'

}

}

});

// создаем шарики

var balls = Composites.pyramid(0, 0, ballPyramidColumns, ballPyramidRows, 0, 0, function(x, y) {

return Bodies.circle(x, y, ballSize, {

render: ballStyle,

friction: ballFriction,

restitution: ballBounciness

});

});

Composite.rotate(balls, Math.PI, Vector.create(225, hexYOffset));

// создаем шестигранники для взаимодействия шаров

var hexes = [];

for (var i = 2; i < 11; i++) {

for (var j = i; j > 0; j--) {

var y = hexYOffset + (hexYSpacing \* i),

x = hexXOffset + ((hexXSpacing \* j) - ((hexXSpacing / 2) \* i)),

size,

style;

if (j == 1 || j == i) {

var direction = (j == 1) ? -1 : 1;

x += direction \* (boundaryHexSize / 4);

y -= 4;

size = boundaryHexSize;

style = transparentStyle;

} else {

size = hexSize;

style = boundaryStyle;

}

hexes.push(Bodies.polygon(x, y, 6, size, {

isStatic: true,

render: style,

chamfer: {

radius: hexChamfer

}

}));

}

};

// создаем столбики для вертикальных границ

var tubes = [];

for (var i = 0; i < 10; i++) {

tubes.push(Bodies.rectangle(174 + (i \* hexXSpacing), 570, 37, 265, {

isStatic: true,

render: transparentStyle

}));

};

// добавляем все созданные обьекты

World.add(engine.world, balls);

World.add(engine.world, hexes);

World.add(engine.world, tubes);

// добавляем стороны воронки для прохода шаров

World.add(engine.world, Bodies.rectangle(280, 10, 15, 250, {

isStatic: true,

angle: Math.PI \* -0.2,

render: transparentStyle

}));

World.add(engine.world, Bodies.rectangle(434, 60, 15, 150, {

isStatic: true,

angle: Math.PI \* 0.2,

render: transparentStyle

}));

// добавляем границы доски

World.add(engine.world, [

// верхняя горизонтальаня граница

Bodies.rectangle(372, 0, 220, borderWidth, {

isStatic: true,

render: transparentStyle

}),

// нижняя горизонтальная граница

Bodies.rectangle(372, height, 430, borderWidth, {

isStatic: true,

render: transparentStyle

})

]);

//отрисовка всех стилей всех обьектов(если убрать то будет черно белое)

var renderOptions = engine.render.options;

renderOptions.wireframes = false;

// запускаем движок

Engine.run(engine);

$('#gravity').change(function(e) {

engine.world.gravity.y = e.target.value;

});

// function stop(){

// Engine.clear(engine);

// };

//добавление функции пепеворота на обьект кнопки

function flip() {

$('canvas').toggleClass('flip');

engine.world.gravity.y = -1 \* engine.world.gravity.y;

};

**Список используемых источников**

1. <https://habrahabr.ru>
2. <http://brm.io/matter-js/docs/>
3. <http://api.jquery.com>
4. <https://ru.wikipedia.org/wiki/Доска_Гальтона>
5. https://intellect.ml/doska-galtona-kvinkunks-dlya-demonstratsii-tsentralnoj-predelnoj-teoremy-4623