Санкт-Петербургский государственный политехнический университет

Факультет технической кибернетики

Кафедра компьютерных систем и программных технологий

Курсовой проект

по Java технологиям:

«Отчет по курсовому проекту»

Выполнил студент группы Воробьев О.О.

Преподаватель Пышкин Е.В.

Санкт-Петербург 2012

**Образ и границы проекта:**

**Формулировка задачи:**

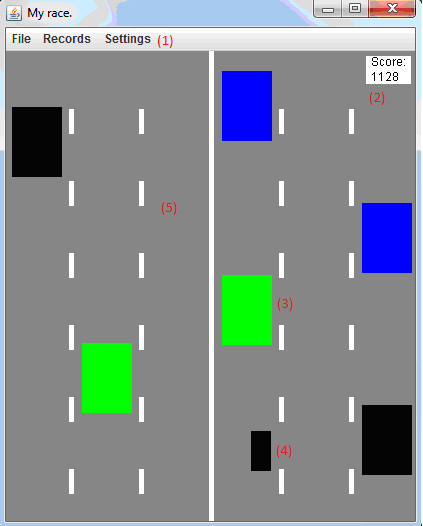
Необходимо реализовать приложение, являющееся интерпретацией классических гонок по вертикали. Необходимо предусмотреть различные уровни сложности игры, подсчет очков, в зависимости от сложности и реализовать таблицу рекордов.

**Структура исходных данных:**

Исходными данными в данном приложении являются рекорды предыдущих игроков. Так как кроме простого массива цифр в файле ничего содержаться не будет, можно ограничиться обычным текстовым файлом что упрощает написание программы.

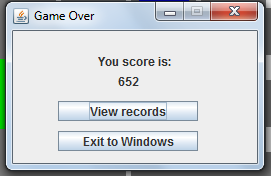
**Основные сценарии функционирования программы:**

Ниже приведен образец главного экрана приложения.



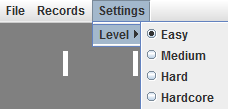
Основная идея приложения следующая: дорога(5) поделена на 2 части 3 полосы встречного движения и 3 «наших» полосы. Про этим полосам вертикально вниз движутся блоки(3) которые имитируют машины. Стоит отметить что машины по встречным полосам движутся быстрее. Задача состоит в уклонении контролируемой машинкой(4) от генерируемых машин. Машинка управляется стрелочками. Уровни сложности регулируют расстояния между препятствиями делая проезд либо сложнее либо проще, в зависимости от выбранного уровня. Подсчет очков осуществляется в реальном времени и выводится на табло(2). Скорость набора очков зависит как от уровня сложности так и от положения машинки на дороге(за езду по встречной дается удвоенное количество очков).

По окончании игры выводится новое окно.



Как видно из картинки, игроку печатается его счет, а так же предлагается 3 варианта: первый- посмотреть рекорды(кнопка “View records”), второй- выйти в Windows (кнопка “Exit to Windows”) и третий- вернуться в игру, закрыв окно.

Так же приложение содержит панель меню(1).



Вкладка «File» содержит пункты: начало игры, пауза/продолжить, выход. Все команды подвешены на хоткеи что облегчает их вызов во время игры.

Вкладка «Records» содержит единственный пункт: посмотреть таблицу рекордов. При нажатии выводится новое окно, содержащее саму таблицу и кнопку выхода из игры.



Вкладка «Settings» позволяет настроить сложность игры.

**Техническое задание:**

Необходимо создать структуру реализующую машинку, препятствие, дорогу, разработать классы представления, алгоритмы поиска столкновений и создания препятствий.

Следует разработать систему подсчета очков.

Должна быть реализована возможность сохранения рекордов и храниться они будут в TXT файле.

**Системная архитектура:**

**Используемые классы:**

Block(класс представляющий препятствие):

x,y-координаты

width,heigth- ширина,высота

color- цвет

dx,dy- координаты векора перемещения блока.

Blockview(класс прорисовывающий объекты класса Block)

Car(класс представляющий машинку):

x,y-координаты

width,heigth- ширина, высота

color- цвет

Carview(класс прорисовывающий объекты класса Car)

Level(класс создающий и рисующий уровень)

MainFrame(класс реализующий главное окно)

MainPanel(класс реализующий панель)

RecordList(класс создающий окно с таблицей рекордов)

Recordmass(класс обрабатывающий массив рекордов и работающий с файлом)

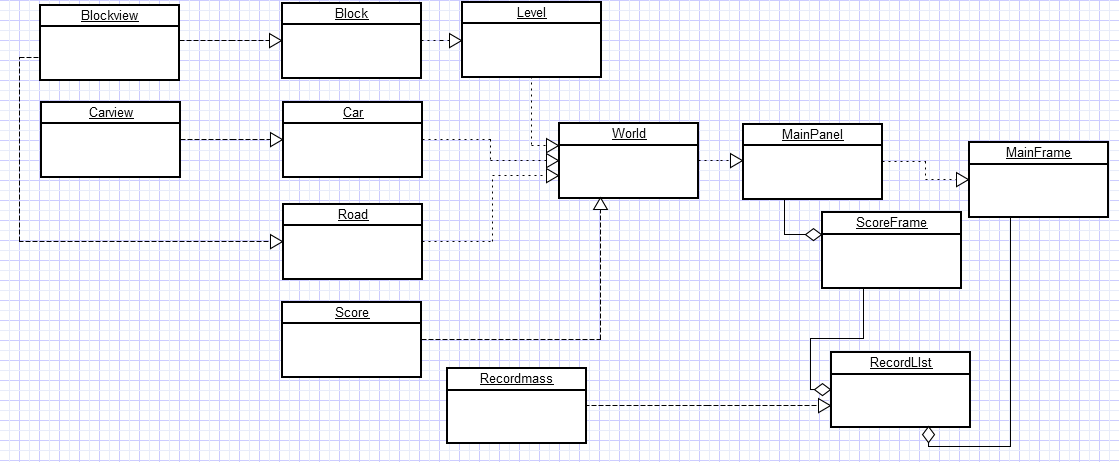
Road(класс создающий и рисующий дорогу и разделительные полосы)

Score(класс отвечающий за подсчет очков)

ScoreFrame(класс создающий окно конца игры)

World(класс объединяющий объекты Road, Level и Car в единый объект)

**Взаимоотношение между классами:**



**Описание выбранного алгоритма создания уровня:**

Алгоритм состоит из 3х функций. Первая(is\_space), проверяет есть ли место между блоками для прохода машинки. Вторая(creation), создает блок одного из 3х цветов на одной из 6ти позиций. Затем проверяет не повторяется ли позиции предыдущих нескольких блоков и созданного. И, если блок проходит еще и проверку функцией is\_space, то его включают в массив блоков уровня. Третья функция двигает весь массив на одну позицию вниз и запускает вторую.

boolean is\_space(Block bl)

{

boolean otv=true;

for(Block block: mass)

{

if ((block.get\_x()-test\_car.get\_width()-10 < bl.get\_x()+bl.get\_width()) &&

(block.get\_x()+block.get\_width()+test\_car.get\_width()+10 > bl.get\_x()) &&

(block.get\_y()-test\_car.get\_heigth()-90-difficultly< bl.get\_y()+bl.get\_heigth()) &&

(block.get\_y()+block.get\_heigth()+test\_car.get\_heigth()+90+difficultly > bl.get\_y()))

{

otv=false;

break;

}

}

return otv;

}

void creation()

{

Block new\_block;

int index=r\_gen1.nextInt(4);

if(index==0) this.color=Color.BLUE;

if(index==1) this.color=Color.GREEN;

if(index==2) this.color=Color.RED;

if(index==2) this.color=Color.BLACK;

int index1= r\_gen1.nextInt(6);

int index2= r\_gen2.nextInt(5);

if(index2 <= ver[index1])

{

if ((index1 != last\_index1) && (index1 != last\_index2) && (index1 != last\_index3))

{

new\_block= new Block(test\_road.get\_x()+6+70\*index1,0,50,70,color);

if ((index1==0) || (index1==1) || (index1 ==2)) new\_block.set\_speed(0, 4);

else new\_block.set\_speed(0,2);

if(is\_space(new\_block))

{

mass.add(new\_block);

last\_index4=last\_index3;

last\_index3=last\_index2;

last\_index2=last\_index1;

last\_index1 = index1;

this.space=0;

}

}

}

}

void level\_step()

{

for(Block block: mass)

{

block.step();

}

creation();

}

**Тестирование приложения:**

**Основные сценарии работы:**

**Вывод:**

В ходе выполнения курсовой работы мы научились использовать язык Java для программирования графических приложений. Также для выполнения поставленной задачи необходимо было спроектировать классы реализующие возможность представления графов в программе. Для этого графа необходимо было выбрать и реализовать алгоритм нахождения кратчайшего пути между двумя любыми узлами.

Самой сложной частью курсовой было реализация алгоритма поиска пути и графическая реализация графа и работы с ним. С чем мы в какой-то степени справились.