Санкт-Петербургский политехнический университет Петра Великого

Кафедра компьютерных систем и программных технологий

**Отчёт по проекту**

**Дисциплина**: Алгоритмы и структуры данных

**Реализация однопользовательского режима для Царской игры Ура**

Выполнил студент гр. 3530901/90003 А.А Ундольский

(подпись)

Преподаватель М.Х. Ахин

(подпись)

“ ” 2020 г.

Санкт-Петербург

2020

Оглавление

[**1. Техническое задание** 3](#_Toc57231376)

[**2. Метод решения** 5](#_Toc57231377)

[**3. Описание классов** 6](#_Toc57231378)

[**4. Работа программы** 9](#_Toc57231379)

**5. Тесты………………………………………………………………………………………………………………….** 8

# **1. Техническое задание**

**Описание приложения**

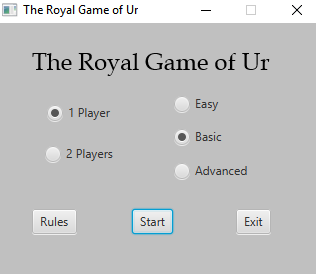
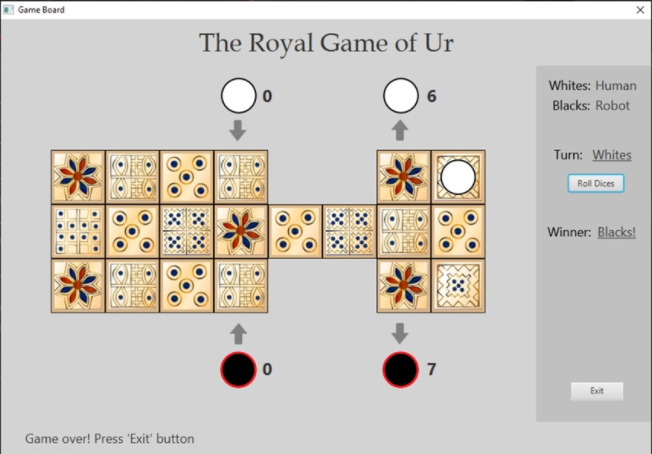
**Общие сведения**

Для Царской игры Ура, реализованной мною в прошлом семестре, необходимо реализовать однопользовательский режим игры, т.е. человека с роботом. При этом робот должен уметь побеждать в большинстве случаев.

**Более подробная информация**

* Приложение будет реализовано на JavaFX
* На стартовом окне приложения есть следующие компоненты:
  + Заголовок с названием игры
  + Кнопки выбора режима игры
  + Кнопки выбора сложности для однопользовательского режима
  + Кнопка, выводящая окно с правилами
  + Кнопка запуска игры
* На окне правил есть следующие компоненты:
* Картинка, демонстрирующая порядок хода
* Текстовое поле с правилами и описанием игры
* Ссылка на видео об игре
* На игровом окне приложения есть следующие компоненты:
  + Кнопка «Roll dices» - позволяет получить случайное число (результат броска костей), на которое игрок должен походить. Поле «Dices» над кнопкой позволяет продемонстрировать это число.
  + Поля «Whites», «Blacks», «Level» - позволяют узнать, кто за кого играет и чей ход.
  + Активные поля доски – те, по которым перемещаются фишки.
  + Поле подсказок – на него выводятся подсказки, что сейчас требуется от игрока
  + Кнопка «Exit» - обеспечивает закрытие окна.

**Предварительный вид приложения**

**GitHub**

https://github.com/tengen-toppa-gurren-lagann/UrGame

# 

# **2. Метод решения**

В основе решения лежит идея, что из-за вероятностного характера событий, происходящих на доске, решать подобную задачу деревьями – трудоёмко и неэффективно. Было принято решение использовать оценочный алгоритм, позволяющий определить ценность хода каждой из фишек и выбрать наилучший.

Алгоритм работает так:

Отслеживаются несколько параметров фишки: ценность клетки, на которую она может пойти, ценность сброса фишки противника, если на вышеупомянутой клетке таковая есть, а также сколько своих и чужих вражеских фишек уже вышло с доски.

Первым делом были прописаны массивы, отвечающие за вышеупомянутые параметры, в которые были занесены соответствующие значения, вычисленные исходя из ценности каждого конкретного действия и подправленные экспериментально. Например, ценность занятия розетки на центральном ряду намного выше, чем ценность хода на первую клетку в том же ряду, так как розетка позволяет совершить повторный ход и, более того, фишка на конкретно данной розетке неуязвима к сбрасыванию.

Далее прописан выбор режима игры, в соответствии с которым выбирается алгоритм, по которому будет осуществляться выбор фишки. Например, в самом простом режиме робот выбирает фишку наугад, как игрок, никогда не игравший в эту игру ранее. В более сложном, учитываются первые два параметра. В самом сложном – все три.

Далее идёт перебор фишек с выбором наиболее ценной для осуществления хода.

# 

# **3. Описание классов**

В классе **Main** содержится main функция программы, загрузчик, задание параметров окна, а также лаунчер приложения.

start() Отвечает за генерацию стартового окна

main() Основная функция программы

Класс **Board Controller** содержит всё необходимое для отрисовки доски, игровой порядок, а также функции взаимодействия с доской.

setGameMode() задаёт режим игры (внутри класса)

setBoardStage() задаёт сцену (внутри класса)

setDifficulty() задаёт сложность (внутри класса)

addPane() Добавить в клетку холст вместе с обработчиком мыши

fillBoardCells() Заполнить доску клетками

initBoard() Инициализация доски

drawBoard() Отрисовка доски

onClickOnBoard() Обработка нажатия мыши на доске

onWhiteChipOnHandBtn() Обработка нажатия на фишку в руке (аналогично и для BlackChip)

onWhiteChipOutBtn() Обработка нажатия на выходное поле (аналогично и для BlackChip)

onExitGameButton() Обработка нажатия на кнопку выхода

onRollDiceButton() Обработка броска кубика

playSound() Воспроизводит звук

processGame() Ведение, смена и обработка этапов игры

onTimer() Таймер для удобозримой визуализации

Класс **Start Controller** отвечает за работу объектов, используемых в главном меню и на странице правил.

setMainStage() Задаёт сцену

onOnePlayerButton() и onTwoPlayersButton() Обрабатывает нажатия на кнопки выбора режима игры

onEasyButton() onBasicButton() onAdvancedButton() Обрабатывают нажатия на кнопки выбора сложности

onStartButton() Обработка нажатия на кнопку Start с выводом окна с игровым полем.

onRulesButton() Обработка нажатия на кнопку Rules c выводом окна с правилами

onRulesOkButton() Обработка нажатия на кнопку ОК на окне с правилами, закрывающее данное окно

onExitButton() Обработка нажатия на кнопку Exit, закрывающее окно (и программу)

onLinkToVideo() Обработка нажатия на ссылку на видео об игре

Класс **Chip** содержит описание параметров фишки и команд для взаимодействия с ней.

getColor() Получить цвет фишки

moveTo() Переместить фишку в указанное место

getRow() Получить ряд, на котором стоит фишка (х)

getCol() Получить столбец, на котором стоит фишка (у)

isOut() Проверка, что фишка вышла с доски

isOnHand() Проверка, что фишка на руке

moveOut() Вывод с доски (дошла до конца)

throwOff() Удаление с доски (побита врагом)

Класс **ChipColor** необходим только для корректной работы enum заданных цветов фишек.

Класс **Dice** представляет собой генератор случайного числа в заданных пределах.

diceRoll() Получаем случайное число, которое может выпасть на костях

Класс **UrBoard** включает в себя основные параметры игры – проверку возможности хода, маршрут движения фишек, получение информации о текущей ситуации на доске, верное перемещение фишек по доске.

getChips() Получить массив нужных фишек

getChipsOnHandCnt() Получить количество фишек на руке

getChipFromHand() Получить фишку с руки

getChipsOutCnt() Получить количество вышедших фишек

getCellForMoveChip() Получить клетку, на которую может переместиться фишка

moveIsPossible() Проверка возможности хода

getChipByPos() Получить фишку, стоящую на данной позиции

getDiceNum() Получить значение на кубике

setDiceNum() Задать значение на кубике

getNewPosition() Получить новую позицию при перемещении фишки на n клеток

moveChipOrCheck() Передвинуть фишку на n клеток или проверить, можно ли передвинуть

Класс **UrBot** представляет из себя робота, определяющего ход компьютера в однопользовательском режиме игры за счёт вычисления лучшего хода для фишки и перебора всех фишек.

setDifficulty() Задать сложность (внутри класса)

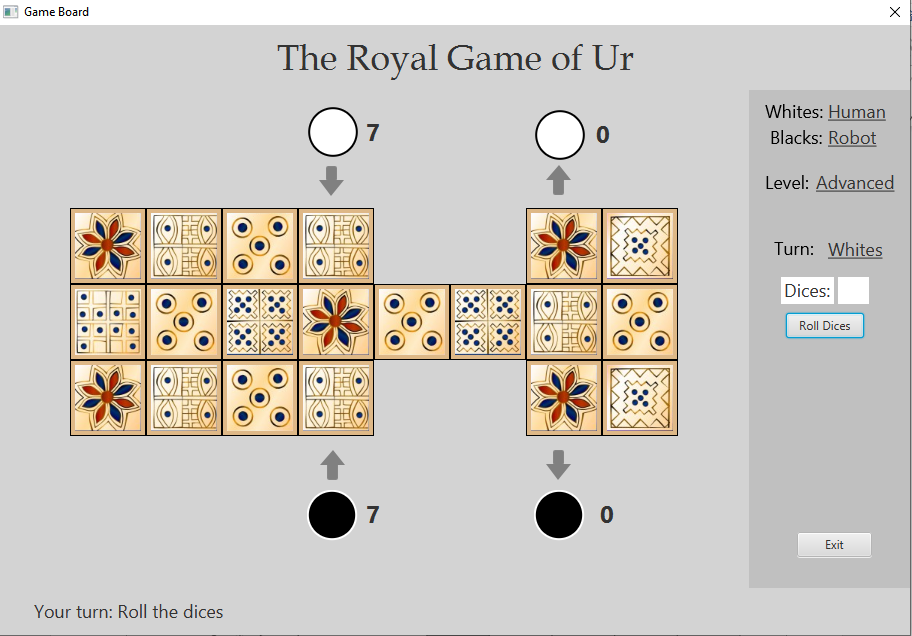
getMoveValue() Целевая Функция

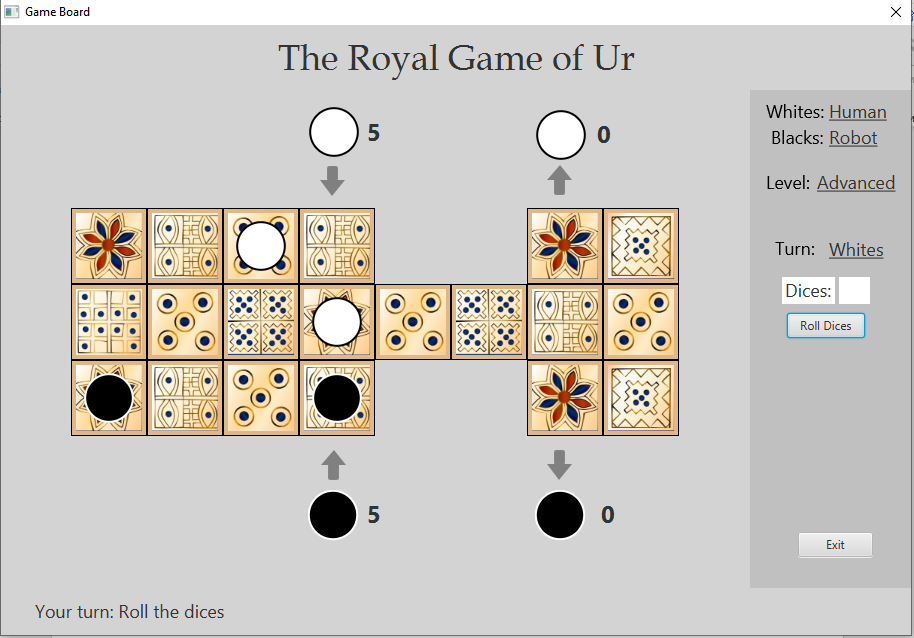
getMostProfitableChip() Получить наиболее ценную для хода фишку

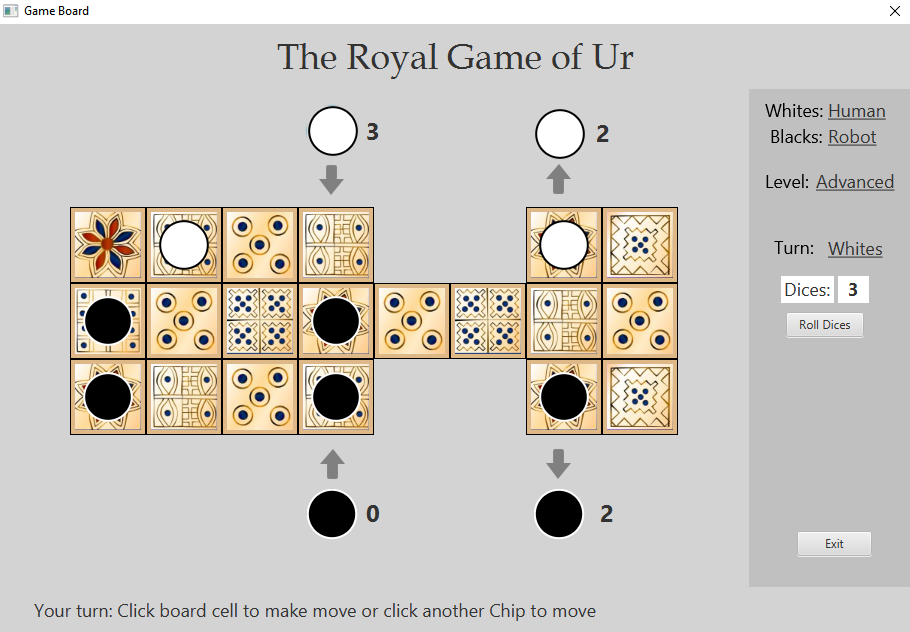
# 

# **4. Работа программы**









**5. Тесты**

