**КАЗАХСКИЙ НАЦИОНАЛЬНЫЙ ПЕДАГОГИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ АБАЯ**

«**MEMORY PUZZLE**»

(проект)

**Выполнила:** Оразбек А. А.

**Специальность:** 6B01509 –

Информатика и робототехника

**Руководитель:** Киселева Е. А.

**Алматы 2022**

**Оглавление**

|  |  |
| --- | --- |
| Паспорт проектной работы | 2 |
| Введение  Основная часть: | 3 |
| Глава 1. Память | 4 |
| * 1. Память. Развититие памяти | 4 |
| 1.2 Игра memory | 6 |
| Глава 2. Описание работы кода | 6 |
| 2.1 Инструкция к мини-игре | 6 |
| 2.2 Разбор кода по частям | 7 |
| Заключение  Код  Список литературы | 10  11  16 |

**Паспорт проектной работы:**

|  |  |
| --- | --- |
| Название проекта | Memory puzzle |
| Участница проекта | Оразбек А. А. |
| Научный руководитель | Киселева Е. А. |
| Цель проекта | Разработать игру, развивающую память |
| Задачи проекта | 1 – создать макет игры  2 – написать код  3 – добавить изображения  4 – продемонстрировать игру |

**Введение:**

**Актуальность проекта.** Память – это важнейшая характеристика психической жизни личности. Без памяти невозможно нормальное функционирование ни личности, ни общества. Человек, приобретает новые знания каждый день. Объем информации, которую нужно запомнить очень велик. Регулярно тренируя память, мы улучшаем общий интеллект, расширяем границы мышления и воображения. Данный проект помогает улучшить память.

**Цель проекта.** Написать код, для развития памяти.

**Задачи проекта.** Для достижения цели, были созданы следующие задачи:

1. Создать макет игры
2. Написать код
3. Добавить изображения
4. Продемонстрировать игру

**Глава 1. Память**

* 1. **Память. Развитие памяти**

Память – это одна из свойств нервной системы, которая заключается в способности какое-то время сохранять информацию о событиях внешнего мира и реакциях организма на эти события, а также многократно воспроизводить и изменять эту информацию.

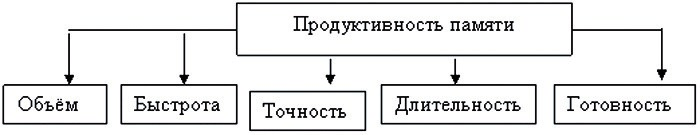


Рис.1 – Продуктивность памяти

Наиболее важные черты, неотъемлемые характеристики памяти - это длительность, быстрота, точность, готовность, объём. От этих характеристик зависит то, насколько продуктивна память человека.

1. Объём - способность одновременно сохранять значительный объём информации. Средний объём памяти - 7 элементов информации.

2. Быстрота запоминания отличается у разных людей. Скорость запоминания можно увеличить с помощью специальной тренировки памяти.

3. Точность проявляется в припоминании фактов и событий, с которыми сталкивался человек, а также в припоминании содержания информации. Эта черта очень важна в обучении.

4. Длительность – способность в течение долгого времени сохранять пережитый опыт. Очень индивидуальное качество: некоторые люди могут вспомнить лица и имена школьных друзей спустя много лет (развита долговременная память), некоторые забывают их спустя всего несколько лет. Длительность памяти имеет выборочный характер.

5. Готовность к воспроизведению - способность быстро воспроизводить в сознании человека информацию. Именно благодаря этой способности мы можем эффективно использовать приобретенный раньше опыт.

С возрастом у людей ухудшается способность к запоминанию, а также появляется рассеянность. Человеку необходимо держать в голове слишком много информации, из-за чего он иногда забывает самые очевидные факты. Причины плохой памяти у взрослых сокрыты как в возрастных изменениях, так и в неправильном образе жизни, стрессах, плохом сне и многом другом.

Память можно натренировать, как, например, мышцу тела. Для этого нужно делать специальные упражнения для памяти. Для того чтобы память продолжала развиваться, человек должен получать впечатления. Хорошо, если происходят различные приятные события, и люди стараются их не забыть.

Существуют правила, позволяющие памяти всегда оставаться в рабочем состоянии: обогащать кровь кислородом, обязательно хорошо высыпаться, не злоупотреблять алкоголем и табаком.

**Методики запоминания от гениев**

Психолог Карл Сишор считает, что обычный человек использует свою память только на 10%, в то время как 90% остаются без применения. Мало кто знает, что практически все методики развития памяти у взрослых базируются на трех природных законах запоминания. Речь идет про эмоции, ассоциации и повторения. Знание этих правил способно помочь как в обыденной жизни, так и в ответственных ситуациях.

Закон эмоций гласит, что **для лучшего запоминания достаточно получить яркие впечатления о заданном предмете.** Самым известным человеком, использовавшим данное правило, был Рузвельт. Он всегда сохранял отличную концентрацию внимания. Все, что он прочитывал, запоминал почти дословно. Секрет данной методики развития памяти у взрослых сокрыт в необходимости полностью сосредоточиться, хотя бы ненадолго, на нужной информации. Именно в этом случае она запомнится лучше, чем если долго размышлять о ней и отвлекаться.

**Удивительную методику оставил после себя Наполеон.** Он отлично помнил на смотре войск расположение каждого своего бойца и его фамилию. Секрет его запоминания имени человека состоял в том, чтобы получить о нем более яркое впечатление. Например, спросив, как пишется его фамилия.

**Президент Линкольн имел свою методику запоминания: он читал вслух то, что было важно помнить.**Получается, что нужно задействовать как можно больше органов чувств. Это позволяет достаточно эффективно повлиять на развитие памяти у взрослых. Упражнения, которые строятся на задействовании нескольких чувств, рекомендуются большинством психологов. К примеру, чтобы запомнить, достаточно записать, а затем мысленно представить написанное.

**Марк Твен часто читал лекции.** Чтобы запомнить длинный текст, он записывал пару слов с начала каждого абзаца. Перед выступлением Твен повторял всю лекцию, используя эту шпаргалку. Но затем ему пришла в голову другая идея - и он стал рисовать то, что ему нужно было запомнить. Таким образом, гении прошлого смогли воплотить все три закона запоминания.

* 1. **Игра memory**

Игра Memory более известна под названием Найди пару, в которую многие из вас наверняка играли, или, по меньшей мере, слышали о ней. Данная игра представляет собой набор из пар одинаковых карточек. Количество пар в наборе начинается, как правило, с двенадцати - версия для самых маленьких. А для тех, кто постарше, карточек больше, обычно раза в два-три.

Суть игры очень простая. Эти карточки выкладываются на стол «рубашкой» вверх. Далее по очереди каждый игрок открывает две любые карточки, показывая их всем. Если на них изображены одинаковые рисунки, он забирает их себе, и вскрывает следующую пару. Однако, если изображения разные, то он возвращает эти карточки в закрытую обратно на свои места, а ход переходит следующему игроку. Когда все карточки будут разобраны, выбирается победитель – тот, кто набрал наибольшее количество карточек.

В memory рекомендуется с 2,5-3 лет и вплоть до самой старости, ведь никому никогда не помешает лишняя тренировка памяти. Впрочем, помимо памяти, игра способствует развитию образного и логического мышления, наблюдательности, мелкой моторики.

Игра memory появилась на свет в феврале 1959 года под предводительством компании Ravensburger, славящейся по всему миру своими пазлами и, отчасти, играми для детей.

**Глава 2. Описание работы кода**

**2.1 – Инструкция к мини-игре**

Инструкция к мини-игре «Memory puzzle»:

1. Запуск игры.
2. Выбор карт: Суть игры заключается в том, чтобы найти пару для карты, которую Вы открыли.
3. Передача управления другому игроку: в том, случае если Вы не смогли найти пару для открытой карты, Вам нужно будет передать управление другому игроку. После нахождения пары на счетчике появится единица, игрок играет до тех пор, пока единицу не заменит число 0, после он должен будет передать управление другому игроку. В случае, если в эту игру играет 1 человек, он должен постараться как можно меньше допускать появление на счетчике числа 0.
4. Выход из игры: Для того, чтобы выйти из игры нажмите на крестик сверху.

**2.2 – Разбор кода по частям**

**1. Класс Cell**

**public class** Cell {  
  
 **public** String **value**;  
 **public int row**;  
 **public int col**;  
 **public boolean wasGuessed**;  
  
 Cell(String value,**int** row,**int** col){  
 **this**.**value** = value;  
 **this**.**row** = row;  
 **this**.**col** = col;  
 **this**.**wasGuessed** = **false**;  
 }  
}

В данной части кода мы создали переменные и конструктор

2. Класс Board

**public void** populateMatrix(){  
  
 **board** = **new** Cell[6][6];  
 String[] images = {**"SmileCat"**, **"DeadCat"**, **"CatMigaushiy"**, **"TwoCat"**, **"CatWithMc"**};  
 Random randomGenerator = **new** Random();  
 **while**(!isBoardFull()){  
 **int** randomImageIndex = randomGenerator.nextInt(images.**length**);  
 String randomImageSelected = images[randomImageIndex];  
  
 **int** randomRow1 = randomGenerator.nextInt(6);  
 **int** randomCol1 = randomGenerator.nextInt(6);  
 **while**(**board**[randomRow1][randomCol1]!=**null**){  
 randomRow1 = randomGenerator.nextInt(6);  
 randomCol1 = randomGenerator.nextInt(6);  
 }  
  
 **int** randomRow2 = randomGenerator.nextInt(6);  
 **int** randomCol2 = randomGenerator.nextInt(6);  
 **while**((randomRow1 ==randomRow2 && randomCol1 == randomCol2)  
 || **board**[randomRow2][randomCol2]!=**null**){  
 randomRow2 = randomGenerator.nextInt(6);  
 randomCol2 = randomGenerator.nextInt(6);  
 }  
  
 **board**[randomRow1][randomCol1] = **new** Cell(randomImageSelected,randomRow1, randomCol1);  
 **board**[randomRow2][randomCol2] = **new** Cell(randomImageSelected,randomRow2, randomCol2);  
  
 }  
  
}

Размер нашей игры 6\*6. В данной части кода мы создали метод, который распределит места наших карточек по индексам. Для это мы создали массив, в котором лежат названия наших карточек. И использовали генератор рандомных чисел, для того, чтобы распределить места карточек.

**private boolean** isBoardFull() {  
 **for**(**int** i=0;i<6;i++){  
 **for**(**int** j=0;j<6;j++){  
 **if**(**board**[i][j]==**null**){  
 **return false**;  
 }  
 }  
 }  
 **return true**;  
}

Этот отрывок кода, нужен для того, чтобы распределить все карты правильно и проверить, когда закончится игра.

3. Класс HelloController

**public void** initialize() **throws** FileNotFoundException {  
   
 **board**.populateMatrix();  
  
 **for** (**int** row = 0; row < 6; row++) {  
 **for** (**int** col = 0; col < 6; col++) {  
 FileInputStream input = **new** FileInputStream(  
 **"src/main/resources/com/example/memorypuzzle/white.jpg"**);  
 Image image = **new** Image(input);  
 ImageView imageView = **new** ImageView(image);  
 imageView.setFitWidth(90);  
 imageView.setFitHeight(90);  
 imageView.setUserData(row+**","**+col);  
 imageView.setOnMouseClicked(event -> {  
 **try** {  
 cardListener(event);  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 });  
 **gameMatrix**.add(imageView, row, col);  
 }  
 }  
  
}

Данный метод, нужен для того, чтобы заполнить места обратной стороной карточек, иначе говоря данный метод заполняет все места картинкой, которую мы задаем.

**public void** cardListener(MouseEvent event) **throws** FileNotFoundException {  
 Node sourceComponent = (Node) event.getSource();  
 String rowAndColumn = (String)sourceComponent.getUserData();  
  
 **int** rowSelected = Integer.*parseInt*(rowAndColumn.split(**","**)[0]);  
 **int** colSelected = Integer.*parseInt*(rowAndColumn.split(**","**)[1]);  
  
 String image = **board**.**board**[rowSelected][colSelected].**value**;  
  
 FileInputStream imageFile = **new** FileInputStream(  
 **"src/main/resources/com/example/memorypuzzle/"**+image+**".jpg"**);  
  
 Image selectedImage = **new** Image(imageFile);  
 ((ImageView)sourceComponent).setImage(selectedImage);  
 checkIfMatchingPairWasFound(rowSelected,colSelected);  
  
}

Благодаря этому методу, карточка переворачивается при нажатии на нее, и мы сможем увидеть картинку.

**public void** checkIfMatchingPairWasFound(**int** rowSelected, **int** colSelected) **throws** FileNotFoundException {  
 **int** score = 0;  
 **if**(**firstCard** == **null**){  
 **firstCard** = **board**.**board**[rowSelected][colSelected];  
 }**else if**(**secondCard** ==**null**){  
 **secondCard** = **board**.**board**[rowSelected][colSelected];  
 }**else** {  
 **if**(**firstCard**.**value**.equals(**secondCard**.**value**)){  
 score = 1;  
 **board**.**board**[**firstCard**.**row**][**firstCard**.**col**].**wasGuessed** = **true**;  
 **board**.**board**[**secondCard**.**row**][**secondCard**.**col**].**wasGuessed** = **true**;  
  
  
 } **else** {  
 **int** indexFirstCardInList = (**firstCard**.**row** \* 6) + **firstCard**.**col**;  
  
 FileInputStream questionFile = **new** FileInputStream(  
 **"src/main/resources/com/example/memorypuzzle/white.jpg"**);  
 Image questionImage = **new** Image(questionFile);  
 ((ImageView)**gameMatrix**.getChildren().get(indexFirstCardInList)).setImage(questionImage);  
  
 **int** indexSecondCardInList = (**secondCard**.**row** \* 6) + **secondCard**.**col**;  
 ((ImageView)**gameMatrix**.getChildren().get(indexSecondCardInList)).setImage(questionImage);  
 }  
  
  
 **firstCard**= **board**.**board**[rowSelected][colSelected];  
 **secondCard** = **null**;  
 **counter**.setText(**"score: "** + score);  
  
 }  
}

Данный метод нужен для того, чтобы оставлять отрытыми карты, пары которых мы нашли, и закрывать карты, пары которых мы не нашли.

**Заключение**

В заключении, хочу написать, что все поставленные выше задачи были выполнены. Нами был написан код игры «memory puzzle», который поможет улучшить память. Мы разобрали что такое память и узнали, как можно ее развивать.

**Код:**

1. Cell.java

**package** com.example.memorypuzzle;  
  
**public class** Cell {  
 *//хранит информацию* **public** String **value**;  
 **public int row**;  
 **public int col**;  
 **public boolean wasGuessed**;  
  
 Cell(String value,**int** row,**int** col){ *//создание конструктора* **this**.**value** = value;  
 **this**.**row** = row;  
 **this**.**col** = col;  
 **this**.**wasGuessed** = **false**;  
 }  
}

1. Board.java

**package** com.example.memorypuzzle;  
  
**import** java.util.Random; *//для распределения рандомных мест используем рандом***public class** Board {  
  
 **public** Cell[][] **board** = **new** Cell[6][6]; *//игра в формате 6\*6* **public void** populateMatrix(){ *//распределяет места карточек* **board** = **new** Cell[6][6];  
 String[] images = {**"SmileCat"**, **"DeadCat"**, **"CatMigaushiy"**, **"TwoCat"**, **"CatWithMc"**}; *//инициализируем матрицу* Random randomGenerator = **new** Random();  
 **while**(!isBoardFull()){ *//пока есть свободные места берем одну картинку и ставим ее на 2 пустых места* **int** randomImageIndex = randomGenerator.nextInt(images.**length**);  
 String randomImageSelected = images[randomImageIndex];  
  
 **int** randomRow1 = randomGenerator.nextInt(6);  
 **int** randomCol1 = randomGenerator.nextInt(6);  
 **while**(**board**[randomRow1][randomCol1]!=**null**){  
 randomRow1 = randomGenerator.nextInt(6); *//рандомный ряд* randomCol1 = randomGenerator.nextInt(6); *//рандомный столбец* }  
  
 **int** randomRow2 = randomGenerator.nextInt(6);  
 **int** randomCol2 = randomGenerator.nextInt(6);  
 **while**((randomRow1 ==randomRow2 && randomCol1 == randomCol2)  
 || **board**[randomRow2][randomCol2]!=**null**){  
 randomRow2 = randomGenerator.nextInt(6);  
 randomCol2 = randomGenerator.nextInt(6);  
 }  
  
 **board**[randomRow1][randomCol1] = **new** Cell(randomImageSelected,randomRow1, randomCol1);  
 **board**[randomRow2][randomCol2] = **new** Cell(randomImageSelected,randomRow2, randomCol2);  
  
 }  
  
 }  
  
 **private boolean** isBoardFull() { *//проверят когда закончится игра, и есть ли пустые места* **for**(**int** i=0;i<6;i++){  
 **for**(**int** j=0;j<6;j++){  
 **if**(**board**[i][j]==**null**){  
 **return false**;  
 }  
 }  
 }  
 **return true**;  
 }  
  
}

1. HelloController.java

**package** com.example.memorypuzzle;  
  
**import** javafx.event.ActionEvent;  
**import** javafx.fxml.FXML;  
**import** javafx.scene.Node;  
**import** javafx.scene.control.Button;  
**import** javafx.scene.control.TextField;  
**import** javafx.scene.image.Image;  
**import** javafx.scene.image.ImageView;  
**import** javafx.scene.input.MouseEvent;  
**import** javafx.scene.layout.GridPane;  
  
**import** java.io.FileInputStream;  
**import** java.io.FileNotFoundException;  
  
**public class** HelloController {  
  
 @FXML  
 **public** GridPane **gameMatrix**;  
 Board **board** = **new** Board();  
 Cell **firstCard** = **null**;  
 Cell **secondCard** = **null**;  
 @FXML  
 **private** TextField **counter**;  
 @FXML  
 **private** Button **button**;  
  
 @FXML  
 **void** onStartClick(ActionEvent event) **throws** FileNotFoundException {  
 initialize();  
  
 }  
  
 @FXML  
 **public void** initialize() **throws** FileNotFoundException {  
 **board**.populateMatrix();  
  
 **for** (**int** row = 0; row < 6; row++) {  
 **for** (**int** col = 0; col < 6; col++) {  
 FileInputStream input = **new** FileInputStream(  
 **"src/main/resources/com/example/memorypuzzle/white.jpg"**);  
 Image image = **new** Image(input);  
 ImageView imageView = **new** ImageView(image);  
 imageView.setFitWidth(90);  
 imageView.setFitHeight(90);  
 imageView.setUserData(row+**","**+col);  
 imageView.setOnMouseClicked(event -> {  
 **try** {  
 cardListener(event);  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 });  
 **gameMatrix**.add(imageView, row, col);  
 }  
 }  
 }  
  
 **public void** cardListener(MouseEvent event) **throws** FileNotFoundException {  
 *//после нажатия на карту, выводится картинка* Node sourceComponent = (Node) event.getSource();  
 String rowAndColumn = (String)sourceComponent.getUserData();  
 System.***out***.println(rowAndColumn); *//выводит на экран местоположение выбранной нами карты* **int** rowSelected = Integer.*parseInt*(rowAndColumn.split(**","**)[0]);  
 **int** colSelected = Integer.*parseInt*(rowAndColumn.split(**","**)[1]);  
  
 String image = **board**.**board**[rowSelected][colSelected].**value**;  
 FileInputStream imageFile = **new** FileInputStream(  
 **"src/main/resources/com/example/memorypuzzle/"**+image+**".jpg"**);  
  
 Image selectedImage = **new** Image(imageFile);  
 ((ImageView)sourceComponent).setImage(selectedImage);  
 checkIfMatchingPairWasFound(rowSelected,colSelected);  
  
  
 }  
  
 **public void** checkIfMatchingPairWasFound(**int** rowSelected, **int** colSelected) **throws** FileNotFoundException {  
 *//метод который оставляет открытыми карьы если они пара, и закрывает если открытые карты не пара* **int** score = 0;  
 **if**(**firstCard** == **null**){  
 **firstCard** = **board**.**board**[rowSelected][colSelected];  
 }**else if**(**secondCard** ==**null**){  
 **secondCard** = **board**.**board**[rowSelected][colSelected];  
 }**else** {  
 **if**(**firstCard**.**value**.equals(**secondCard**.**value**)){  
  
 score = 1;  
 **board**.**board**[**firstCard**.**row**][**firstCard**.**col**].**wasGuessed** = **true**;  
 **board**.**board**[**secondCard**.**row**][**secondCard**.**col**].**wasGuessed** = **true**;  
  
 } **else** {  
 **int** indexFirstCardInList = (**firstCard**.**row** \* 6) + **firstCard**.**col**;  
 FileInputStream questionFile = **new** FileInputStream(  
 **"src/main/resources/com/example/memorypuzzle/white.jpg"**);  
 Image questionImage = **new** Image(questionFile);  
 ((ImageView)**gameMatrix**.getChildren().get(indexFirstCardInList)).setImage(questionImage);  
  
 **int** indexSecondCardInList = (**secondCard**.**row** \* 6) + **secondCard**.**col**;  
 ((ImageView)**gameMatrix**.getChildren().get(indexSecondCardInList)).setImage(questionImage);  
 }  
  
 **firstCard**= **board**.**board**[rowSelected][colSelected];  
 **secondCard** = **null**;  
 **counter**.setText(**"score: "** + score);  
  
 }  
 }  
}

1. HelloApplication.java

**package** com.example.memorypuzzle;  
  
**import** javafx.application.Application;  
**import** javafx.fxml.FXML;  
**import** javafx.fxml.FXMLLoader;  
**import** javafx.scene.Parent;  
**import** javafx.scene.Scene;  
**import** javafx.stage.Stage;  
**import** java.io.File;  
**import** java.io.IOException;  
**import** java.net.URL;  
  
**public class** HelloApplication **extends** Application{  
  
 @Override  
 **public void** start(Stage stage) **throws** IOException {  
 URL url = **new** File(**"src/main/resources/com/example/memorypuzzle/hello-view.fxml"**).toURI().toURL();  
 Parent root = FXMLLoader.*load*(url);  
 stage.setTitle(**"Memory Game"**);  
 stage.setScene(**new** Scene(root));  
  
 stage.show(); *// IMPORTANT* }  
  
 **public static void** main(String[] args) {  
 *launch*();  
 }  
  
}

1. Hello-view.fxml

*<?***xml version="1.0" encoding="UTF-8"***?>  
  
<?***import javafx.geometry.Insets***?>  
<?***import javafx.scene.control.Button***?>  
<?***import javafx.scene.control.TextField***?>  
<?***import javafx.scene.effect.DropShadow***?>  
<?***import javafx.scene.layout.AnchorPane***?>  
<?***import javafx.scene.layout.BorderPane***?>  
<?***import javafx.scene.layout.GridPane***?>  
<?***import javafx.scene.layout.HBox***?>  
<?***import javafx.scene.layout.Pane***?>*<**AnchorPane style="-fx-background-color: transparent;" xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.example.memorypuzzle.HelloController"**>  
 <**children**>  
 <**AnchorPane fx:id="root" minHeight="373.0" minWidth="439.0" AnchorPane.bottomAnchor="5.0" AnchorPane.leftAnchor="5.0" AnchorPane.rightAnchor="5.0" AnchorPane.topAnchor="5.0"**>  
 <**children**>  
 <**BorderPane fx:id="body" styleClass="body" AnchorPane.bottomAnchor="2.0" AnchorPane.leftAnchor="2.0" AnchorPane.rightAnchor="2.0" AnchorPane.topAnchor="2.0"**>  
 <**top**>  
 <**HBox fx:id="menuPanel"**>  
 <**Button fx:id="button" onAction="#onStartClick" text="Start Again"** />  
 <**Pane fx:id="handPaneMac" prefHeight="14.0" prefWidth="295.0"** />  
 <**TextField fx:id="counter" prefHeight="14.0" prefWidth="79.0" promptText="Counter: "** />  
 </**HBox**>  
 </**top**>  
 <**center**>  
 <**HBox**>  
 <**GridPane fx:id="gameMatrix" alignment="center" hgap="10" vgap="10" xmlns:fx="http://javafx.com/fxml"**>  
 <**padding**><**Insets bottom="25" left="25" right="25" top="25"** /></**padding**>  
  
 </**GridPane**>  
 </**HBox**>  
 </**center**>  
 </**BorderPane**>  
 </**children**>  
 </**AnchorPane**>  
 </**children**>  
 <**effect**>  
 <**DropShadow blurType="GAUSSIAN"** />  
 </**effect**>  
</**AnchorPane**>

**Список литературы:**

1 - <http://www.klinika56.ru/paczientam/stati/pamyat-prichinyi-ee-narushenij-i-metodyi-trenirovki-pamyati.html>

1. - <https://works.doklad.ru/view/DR2ngOkHj08.html>