СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc26788356)

[ВВЕДЕНИЕ 4](#_Toc26788357)

[1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ 5](#_Toc26788358)

[2 ПРОГРАММНОЕ ПРОЕКТИРОВАНИЕ 6](#_Toc26788359)

[3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 10](#_Toc26788360)

[3.1 Детальная реализация функциональных частей ПО 10](#_Toc26788361)

[3.2 Сопроводительная документация 19](#_Toc26788362)

[3.3 Анализ ПО 19](#_Toc26788363)

[3.4 Тестирование ПО 20](#_Toc26788364)

[ЗАКЛЮЧЕНИЕ 25](#_Toc26788365)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc26788366)

[ПРИЛОЖЕНИЯ 26](#_Toc26788367)

ВВЕДЕНИЕ

Темой данного курсового проекта является создание мобильного игрового приложения «Морской бой».

Ни для кого не секрет, что видео игры прочно заняли свою позицию в современной индустрии развлечений. В настоящее время в индустрии видеоигр направление мобильных платформ получило активное развитие. Наиболее популярные игры по аудитории значительно опережают игры с ПК и игровых консолей, а по данным компании SuperData [1] на рынке мобильных игр доход за 2016 год составил $40.6 млрд. При общих для индустрии 91$ млрд.

Разработка игровых приложений для мобильных платформ представляет собой наименее сложный, затратный и рискованный способ для разработчика получить опыт в разработке игр. Мобильные игры по структуре значительно проще аналогов с других платформ, а цена размещения готового продукта в таких системах цифрового распространения, как Google Play [2], чаще всего ограничивается разовым взносом регистрации разработчика.

Мобильные игры так же предоставляют разработчику множество способов развития проекта, в том числе распространение уже получившей некоторую известность игры на персональные компьютеры. Такое расширение можно увидеть на примерах игр Braveland [3] или Warhammer 40,000: Deathwatch [4], которые в настоящее время размещены в системе цифрового распространения Steam [5].

По всем вышеприведённым доводам мною было принято решение начать разработку мобильного игрового приложения. Система Android была выбрана по причине её максимальной среди мобильных платформ простоте и открытости, но это не ограничивает проект и в процессе дальнейшего развития также возможно распространение на iOS и Windows Phone.

1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ

Игра, о которой пойдет речь, черпает свои истоки из далеких 1950-х годов, когда были популярны игры на листике среди школьников и студентов. В данном мобильном приложении я постарался реализовать тот старый и знаменитый морской бой, в который можно было сыграть на обыкновенном листике. На сегодняшний день листик заменяет смартфон, который есть у каждого второго учащегося.

Морской бой несет, в себе одновременно простой и глубокий смысл, в одно время ты просто нажимаешь кнопки и пытаешься выиграть, получаешь удовольствие от самого процесса и качества игры, но в тоже время ты сможешь окунуться в свои школьные годы и вспомнить как с соседом по парте соревновались между собой.

Об актуальности данного приложения не стоит беспокоиться, так как спрос на примитивные мобильные приложения, растет быстрыми шагами и не собирает останавливаться.

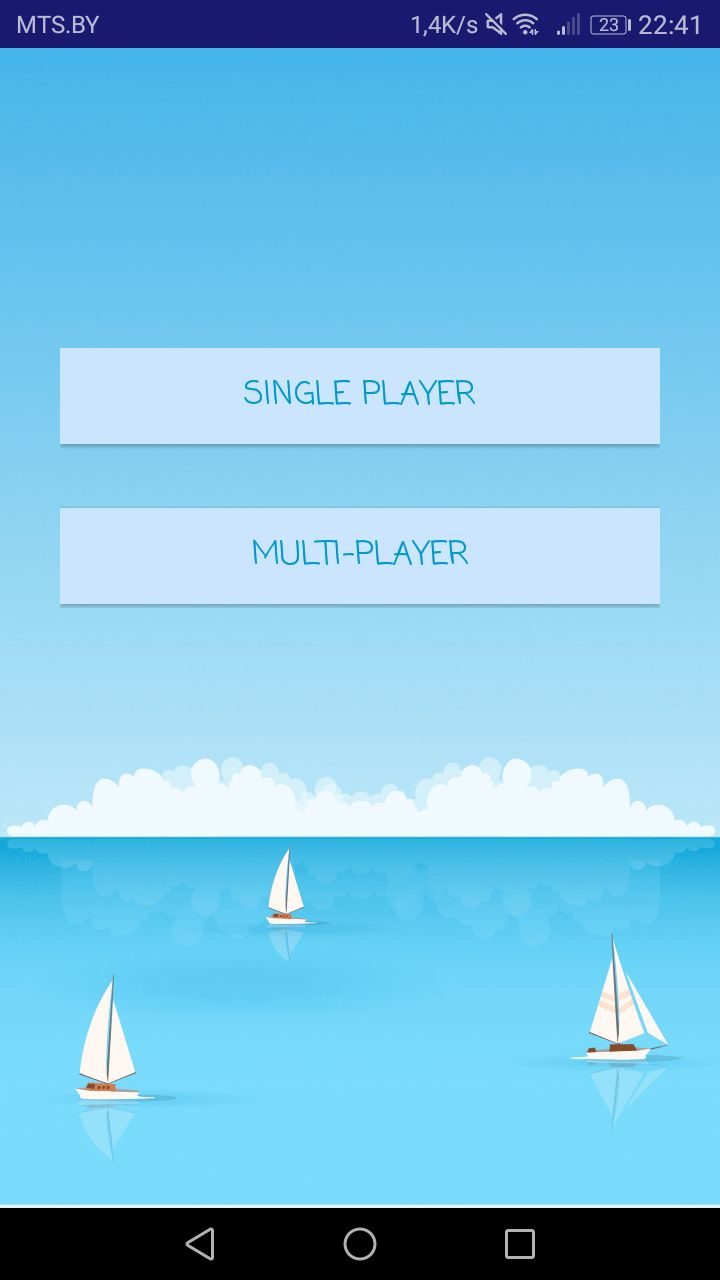
Для реализации своего игрового мобильного приложения я использовал интегрированную среду разработки Android Studio. Android Studio – официальное средство разработки Android приложений. Данная среда разработки доступна для Windows, macOS и Linux. 17 мая 2017, на ежегодной конференции Google I/O, Google анонсировал поддержку языка Kotlin, используемого в Android Studio, как официального языка программирования для платформы Android в дополнении к Java и С++. Andriod Studio имеет расширенный редактор макетов: WYSIWYG, способность работать с UI компонентами при помощи Drag-and-Drop, функция предпросмотра макета на нескольких конфигурациях экрана. Сборка Проекта в Android Studio основана на Gradle. Минимальным Android API был выбран 19, так как он поддерживается у 90,1% пользователей устройств с поддержкой Android. Для написания скриптов на языке Java использовалось Android Studio.

2 ПРОГРАММНОЕ ПРОЕКТИРОВАНИЕ

В рамках данного курсового проекта, будет разработано игровое мобильное приложение «Морской бой».

Так как данное приложение будет разрабатываться в Android Studio, изначально надо будет ознакомиться с официальной документации, подчеркнуть основные моменты насчет игрового движка, почитать информацию об основных библиотеках, которые будут использоваться во время разработки и можно приступать к разработке самой игры.

В соответствии с требованиями, должно быть игровое меню, в котором будет возможность выбрать игровой режим. Вид главного игрового меню представлен на рисунке 2.1.

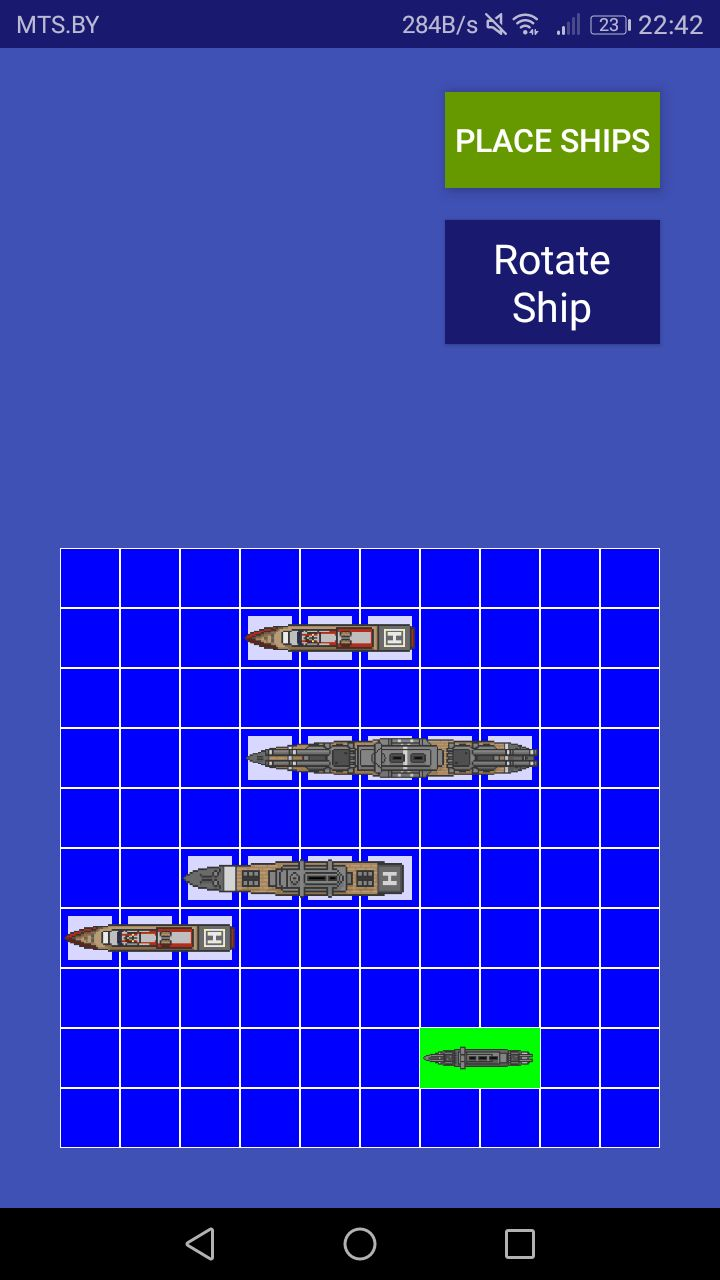


**Рисунок 2.1** – Главное игровое меню

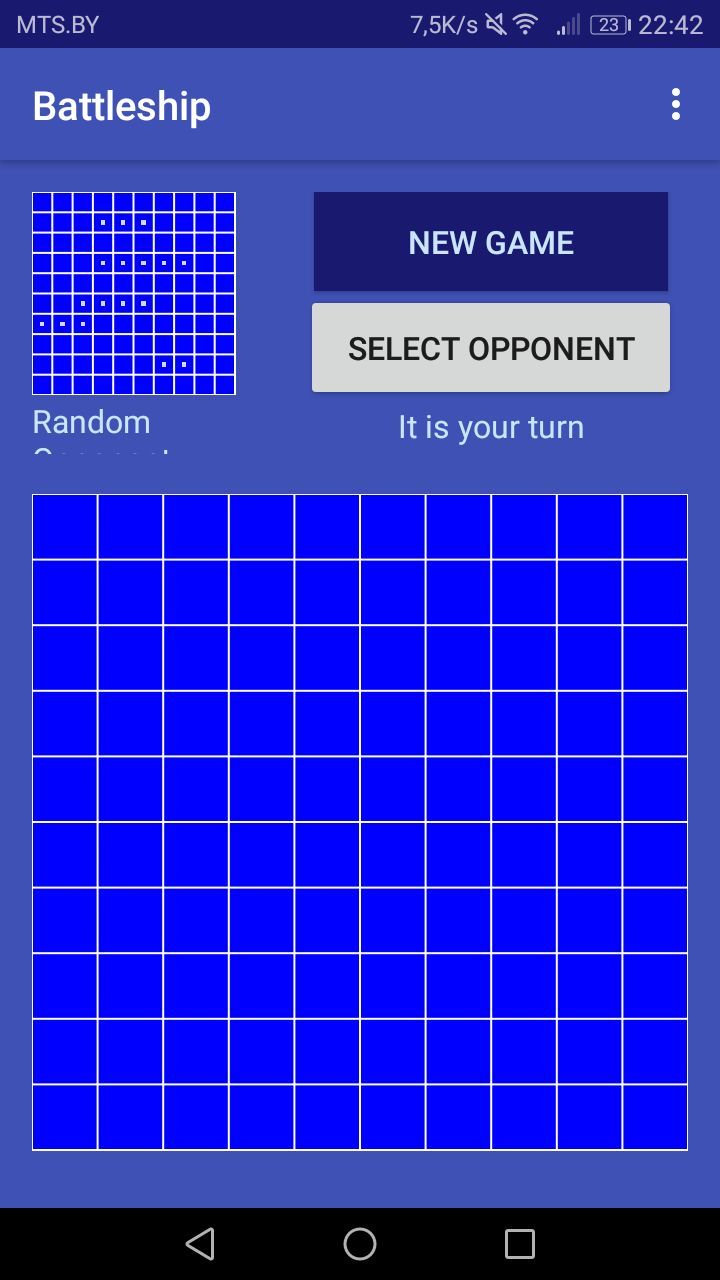
В главном меню предоставляется выбор между режимами, офлайн или онлайн. При выборе режима офлайн, пользователю предоставляется возможность расставить корабли на поле боя. По готовности пользователь сможет перейти к самому бою против бота. В окне боя, пользователь может выбрать сложность противника. Примерный вид данных окон представлен на рисунках 2.2, 2.3 и 2.4:



**Рисунок 2.2 –** Расставление кораблей в офлайн режиме

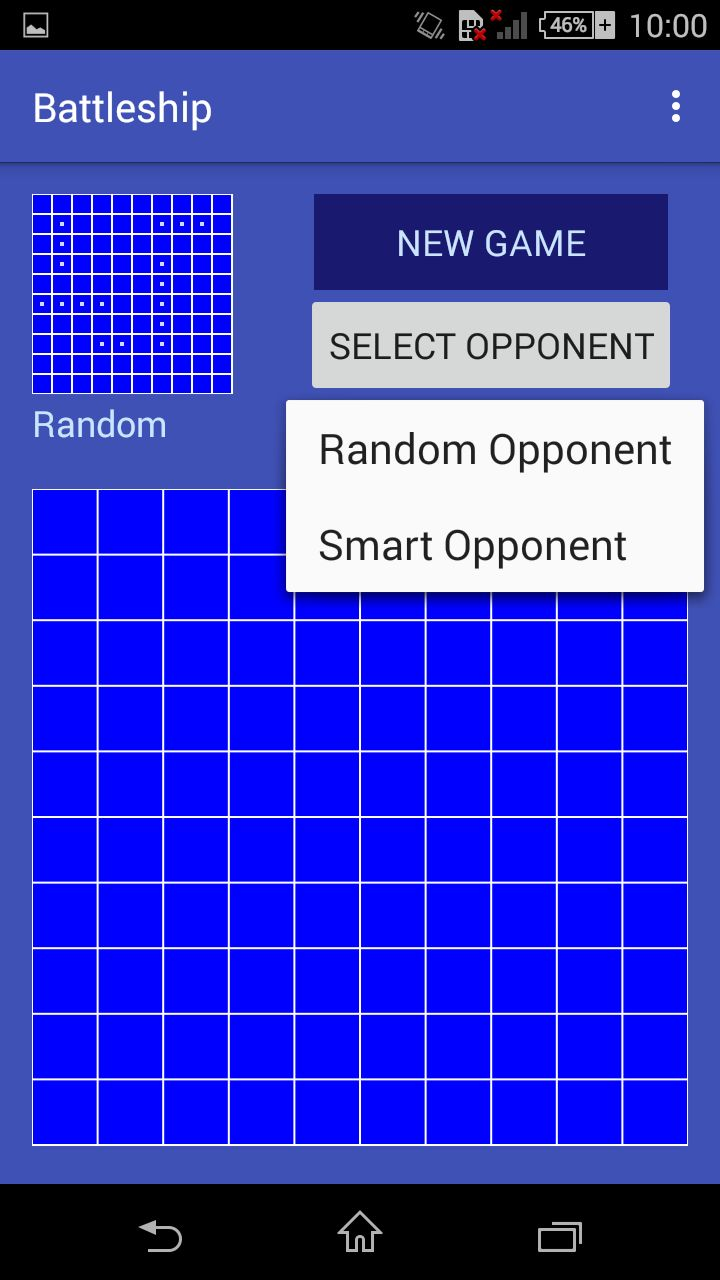


**Рисунок 2.3 –** Готов к бою



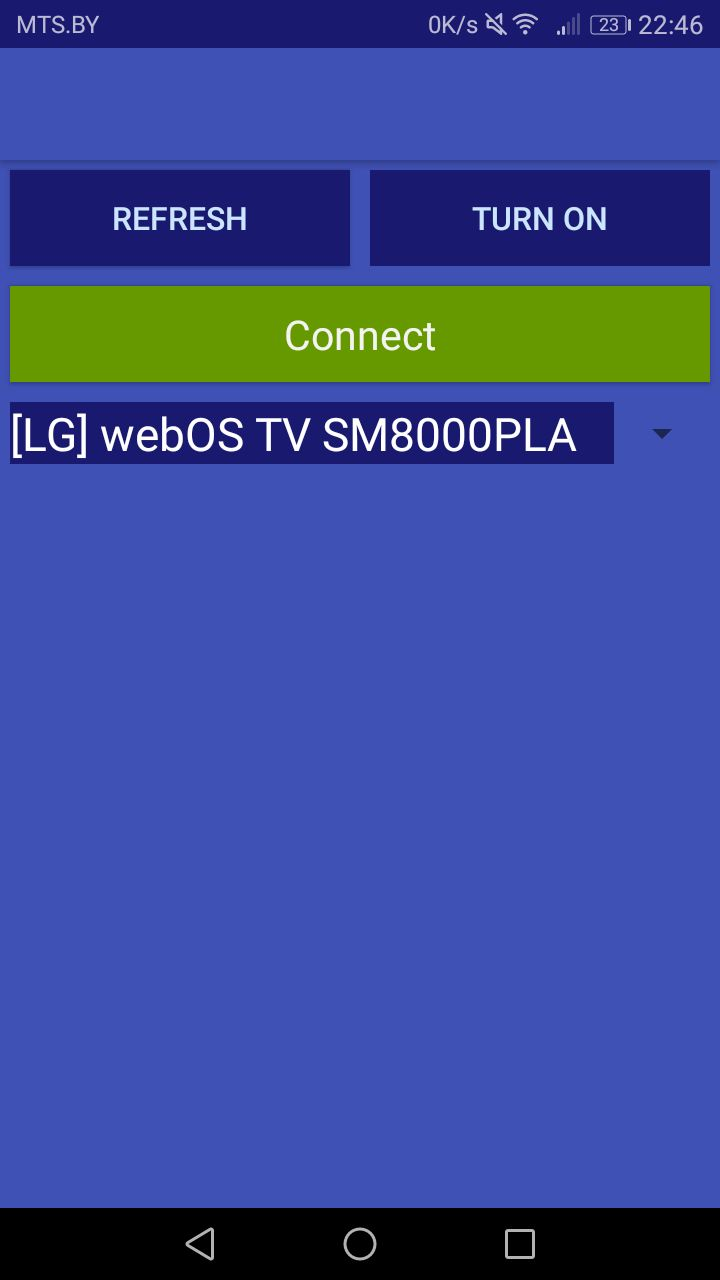
**Рисунок 2.4** – Окно боя с противником

Режим боя против боя подразумевает бой с не настоящим противником. В данном режиме пользователь может ознакомиться с приложением и отточить свои навыки, чтобы сразиться с реальным игроком. Также пользователю предоставляется возможность выбрать уровень бота. Примерный вид изображен на рисунке 2.5.



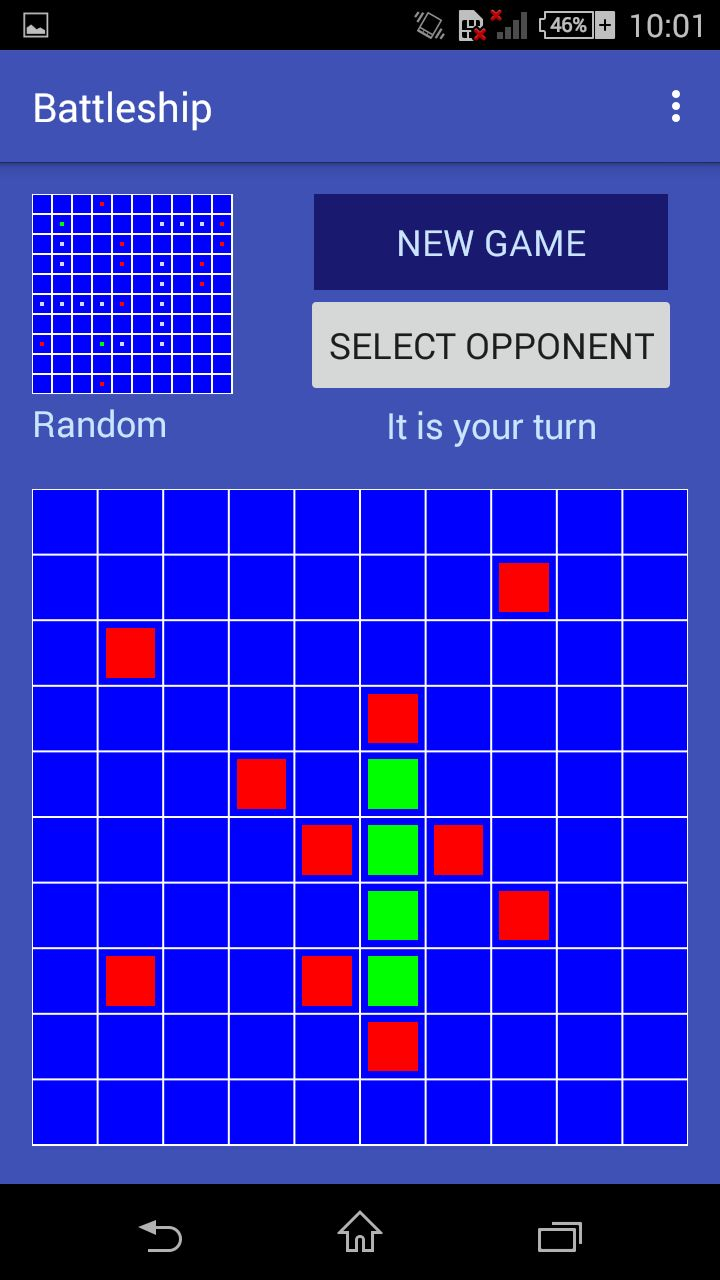
**Рисунок 2.5** – Выбор режима бота

Второй пункт главного меню открывает окно подключения для боя с реальным противником. В данном режиме вы должны выбрать противника и бросить ему вызов. После этого открывается поля для расставления кораблей. Примерный вид изображен на рисунке 2.6.



**Рисунок 2.6** – Окно для вызова настоящего противника

Режим боя против реального противника аналогичен бою против бота, только с реальным противником. Для подключения к бою пользователи должны быть в одной сети Wi-Fi. Примерные боя изображен на рисунках 2.7



**Рисунок 2.7** – Режим боя с реальным противником

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

3.1 Детальная реализация функциональных частей ПО

Главной составляющей курсового проекта является, реализация игры против бота, а также игры онлайн против противника. В самом начале я приступил к прорисовке игральной доски, а также оформления UI.

Изначала я начал прорисовывать доску, на которой будут производиться боевые действия. Также я определился с размерами и визуальным оформлением, в листинге 3.1, будет приведена инициализация доски.

**Листинг 3.1** – Метод createBoard()

1. private void createBoard(Place[][] board){
2. for(int y = 0; y < board.length; y++){
3. for(int x = 0; x < board[0].length; x++){
4. board[y][x] = new Place(x, y);
5. }
6. }
7. }

В листинге 3.2, будет приведен метод, который отвечает за прорисовку, размеры и цвет доски.

**Листинг 3.2** – Метод onDraw()

1. protected void onDraw(Canvas canvas) {
2. super.onDraw(canvas);
3. drawGrid(canvas);
4. drawShotPlaces(canvas);
5. if(displayShips){
6. drawShips(canvas);
7. }
8. drawShipHitPlaces(canvas);
9. }

Далее я начал разрабатывать расстановку кораблей. В данном методе я прохожу по борту и расставляю корабли. В нем исключается возможность наложения одного корабля на другой. Но так как пользователь может расставлять корабли как горизонтально, так и вертикально, я написал метод для поворота корабля. Также мне необходим метод для удаления корабля.

В листинге 3.3, 3.4, 3.5 представлены методы расстановки, поворота и удаления кораблей.

**Листинг 3.3** – Метод placeShip()

1. boolean placeShip(Ship ship, int x, int y, boolean dir){
2. if(ship == null){
3. return false;
4. }
5. removeShip(ship);
6. List<Place> shipPlaces = new ArrayList<Place>();
7. Place place;
8. for(int i = 0; i < ship.getSize(); i++){
9. if(dir){
10. place = placeAt(x+i, y);
11. }
12. else{
13. place = placeAt(x, y+i);
14. }
15. if(place == null || place.hasShip()) {
16. return false;
17. }
18. shipPlaces.add(place);
19. }
20. for(Place placeWithShip: shipPlaces){
21. placeWithShip.setShip(ship);
22. }
23. ship.setDir(dir);
24. ship.placeShip(shipPlaces);
25. return true;
26. }

**Листинг 3.4** – Метод removeShip()

1. private void removeShip(Ship ship){
2. for(int i = 0; i < board.length; i++){
3. for(int j = 0; j < board[0].length; j++){
4. if(board[i][j].hasShip(ship)){
5. board[i][j].removeShip();
6. }
7. }
8. }
9. ship.removeShip();
10. }

**Листинг 3.5** – Метод rotateShip ()

1. private void rotateShip(ShipView shipToRotate) {
2. if (shipToRotate.getShip().getDir()) {
3. shipToRotate.getShipImage().setRotation(90);
4. shipToRotate.getShip().setDir(false);
5. } else {
6. shipToRotate.getShipImage().setRotation(0);
7. shipToRotate.getShip().setDir(true);
8. }
9. }

Одним из основных методов является bourdToched, который отвечает за реагирование на попадание или промах при выстреле. Данный метод будет представлен в листинге 3.6.

**Листинг 3.6**–Метод boardTouched()

1. public void boardTouched(final int x, final int y) {
2. Place placeToHit = game.getOpponentPlayer().getBoard().placeAt(x, y);
3. boolean isGameOver = game.getPlayer().areAllShipsSunk() || game.getOpponentPlayer().areAllShipsSunk();
4. boolean computersTurn = game.getActivePlayer() != game.getPlayer();
5. boolean placeAlreadyHit = placeToHit.isHit();
6. if (isGameOver || computersTurn || placeAlreadyHit) {
7. return;
8. }
9. game.hitPlace(x, y);
10. if (placeToHit.hasShip()) {
11. playSound(R.raw.shiphit);
12. }
13. else {
14. game.changeTurn();
15. playSound(R.raw.miss);
16. }
17. updateTurnDisplay();
18. updateBoards();
19. boolean playerWon = game.getOpponentPlayer().areAllShipsSunk();
20. new Thread(new Runnable() {
21. @Override
22. public void run() {
23. if (NetworkAdapter.hasConnection()) {
24. Log.d("wifiMe", "Wrote message to opponent for placing shot");
25. NetworkAdapter.writePlaceShotMessage(x, y);
26. }
27. }
28. }).start();
29. if (playerWon) {
30. updateWinDisplay(true);
31. resultsDialog(true, game.getShipsSunkCount(game.getPlayer()));
32. return;
33. }
34. if (!NetworkAdapter.hasConnection()) {
35. Log.d("wifiMe", "Computer made a play");
36. boolean isComputersTurn = game.getActivePlayer() != game.getPlayer();
37. if (isComputersTurn) {
38. computerTurn();
39. }
40. }
41. }

После прорисовки доски и обработки нажатий на экран для выстрелов, я начал писать бота. В моем приложении есть способность менять уровень сложности бота. Исходя из этого я написал два разных по стилю боя алгоритма для ботов.

Стратегия для рандомного и умного ботов будут представлены в листинге 3.7 и 3.8.

**Листинг 3.7** – Метод pickStrategyMove()

1. public Place pickStrategyMove(Board board) {
2. if (board == null || board.isAllHit()) {
3. return null;
4. }
5. Random rng = new Random();
6. int boardSize = board.size();
7. Place toHit = null;
8. while (toHit == null || toHit.isHit()){
9. toHit = board.placeAt(rng.nextInt(boardSize), rng.nextInt(boardSize));
10. }
11. return toHit;
12. }

**Листинг 3.8** – Метод target()

1. private Place target(Board board){
2. int lastX = lastMoveHitShip[0];
3. int lastY = lastMoveHitShip[1];
4. Place toHit = null;
5. if(directionsToTry.size() == 0){
6. toHuntMode();
7. return hunt(board);
8. }
9. int randomDirection = directionsToTry.get(rng.nextInt(directionsToTry.size()));
10. lastDirectionTried = randomDirection;
11. directionsToTry.remove(Integer.valueOf(randomDirection));
12. switch (randomDirection){
13. case UP: toHit = goDirection(board, lastX, lastY, 0, -1); break;
14. case RIGHT: toHit = goDirection(board, lastX, lastY, 1, 0); break;
15. case DOWN: toHit = goDirection(board, lastX, lastY, 0 , 1); break;
16. case LEFT: toHit = goDirection(board, lastX, lastY, -1, 0); break;
17. }
18. if(toHit == null){
19. return target(board);
20. }
21. return toHit;
22. }

Умный бот не стреляет в рандомные места. Бот в моем алгоритме стреляет по принципу шахматной доски, так как мы знаем, что размер корабля не меньше 2-х клеток. Исходя их этого, при проходе, как по шахматной доске, мы увеличиваем шанс победы в двое.

Метод проверки незатронутых клеток будет представлен в листинге 3.9.

**Листинг 3.9** – Метод hunt()

1. private Place hunt(Board board){
2. lastDirectionTried = UNKNOWN;
3. int boardSize = board.size();
4. Place toHit = null;
5. for (int i = 0; i < (boardSize\*boardSize\*4) && (toHit == null || toHit.isHit() || !isCheckerboardPlace(toHit)); i++){
6. toHit = board.placeAt(rng.nextInt(boardSize), rng.nextInt(boardSize));
7. }
8. if(toHit == null || toHit.isHit() || !isCheckerboardPlace(toHit)){
9. for(int i = 0; i < (boardSize\*boardSize\*4) && (toHit == null || toHit.isHit()); i++ ){
10. toHit = board.placeAt(rng.nextInt(boardSize), rng.nextInt(boardSize));
11. }
12. }
13. return toHit;
14. }

В самом конце я начал заниматься оформлением внешнего вида приложения. В следующих листингах я отображу XML код для оформления главного меню и окна в режиме боя.

В листинге 3.10, 3.11 будет предоставлен XML код оформления окон.

**Листинг 3.10** – XML код для оформления главного меню.

1. <LinearLayout
2. android:layout\_width="wrap\_content"
3. android:layout\_height="130dp"
4. android:layout\_marginBottom="150dp"
5. android:orientation="vertical">
6. <Button
7. android:id="@+id/single"
8. android:layout\_width="300dp"
9. android:layout\_height="wrap\_content"
10. android:layout\_marginBottom="@dimen/activity\_vertical\_margin"
11. android:background="@color/colorBlueLight"
12. android:fontFamily="casual"
13. android:textColor="@android:color/holo\_blue\_dark"
14. android:text="Single Player" />
15. <Button
16. android:id="@+id/multi"
17. android:layout\_width="300dp"
18. android:layout\_height="wrap\_content"
19. android:layout\_marginTop="@dimen/activity\_vertical\_margin"
20. android:background="@color/colorBlueLight"
21. android:fontFamily="casual"
22. android:textColor="@android:color/holo\_blue\_dark"
23. android:text="Multi-player" />
24. </LinearLayout>

**Листинг 3.11** – XML код для оформления окна в режиме боя.

1. <LinearLayout
2. android:layout\_width="match\_parent"
3. android:layout\_height="match\_parent"
4. android:orientation="vertical">
5. <LinearLayout
6. android:layout\_width="match\_parent"
7. android:layout\_height="0dp"
8. android:layout\_weight=".25"
9. android:elevation="1dp"
10. android:orientation="horizontal">
11. <LinearLayout
12. android:layout\_width="0dp"
13. android:layout\_height="match\_parent"
14. android:layout\_weight="0.4"
15. android:orientation="vertical">
16. <edu.utep.cs4330.battleship.BoardView
17. android:id="@+id/playerBoardView"
18. android:layout\_width="110dp"
19. android:layout\_height="110dp"
20. android:layout\_weight="0.30" />
21. <TextView
22. android:id="@+id/strategy\_description"
23. android:layout\_width="match\_parent"
24. android:layout\_height="38dp"
25. android:layout\_weight="0.30"
26. android:paddingTop="2dp"
27. android:text="@string/strategy"
28. android:textColor="@color/colorBlueLight" />
29. </LinearLayout>
30. <LinearLayout
31. android:layout\_width="0dp"
32. android:layout\_height="match\_parent"
33. android:layout\_weight="0.6"
34. android:orientation="vertical">
35. <Button
36. android:id="@+id/resetGame"
37. style="@style/Widget.AppCompat.Button.Colored"
38. android:layout\_width="match\_parent"
39. android:layout\_height="0dp"
40. android:layout\_marginLeft="10dp"
41. android:layout\_marginRight="10dp"
42. android:layout\_weight="0.22"
43. android:background="@color/colorPrimaryDark"
44. android:onClick="resetGame"
45. android:text="@string/new\_button\_label"
46. android:textColor="@color/colorBlueLight"
47. android:theme="@style/AppTheme" />
48. <Button
49. android:id="@+id/opponentSelect"
50. android:layout\_width="match\_parent"
51. android:layout\_height="0dp"
52. android:layout\_marginLeft="5dp"
53. android:layout\_marginRight="5dp"
54. android:layout\_weight="0.25"
55. android:onClick="showOpponentSelectPopup"
56. android:text="@string/opponent\_select" />
57. <TextView
58. android:id="@+id/gameStatus"
59. android:layout\_width="match\_parent"
60. android:layout\_height="0dp"
61. android:layout\_weight="0.11"
62. android:text="@string/player\_turn\_status"
63. android:textAlignment="center"
64. android:textColor="@color/colorBlueLight"/>
65. </LinearLayout>
66. </LinearLayout>

3.2 Сопроводительная документация

Сопроводительная документация по разработанному программному продукту предоставляется в составе технического задания (приложение А) согласно ГОСТ 19.201-78.

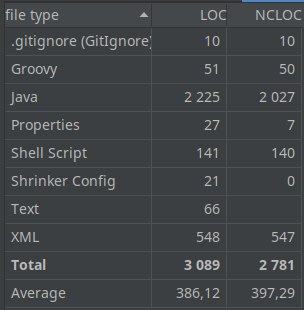
Требования к сопроводительной документации устанавливаются государственными стандартами ЕСПД.

3.3 Анализ ПО

Для анализа данного программного обеспечения используем анализ метрик кода.

Метрика программного обеспечения – мера, позволяющая получить численное значение некоторого свойства программного обеспечения или его спецификаций [4].

В Android Studio нет встроенной функции для анализа кода, поэтому я поставил расширение из плагинов. При анализе метрик будем учитывать количество строк со всего проекта. В данной метирке показывается количество сток на разных языках. Также демонстрируется среднее количество сток исходя из разных языков. В поле NCLOC – подсчитывается количество строк чистого кода, без пробелов и табуляций.



**Рисунок 3.1** — Результат метрик кода

Количество сток для языков: Groovy –51, Java – 2225, XML –548. Количество сток выделенных под свойства – 27. Количество сток с текстом – 66. Всего строк – 3089. Среднее количество сток – 386,12. Число строк чистого кода – 2781.

В целом результаты метрик кода являются удовлетворительными.

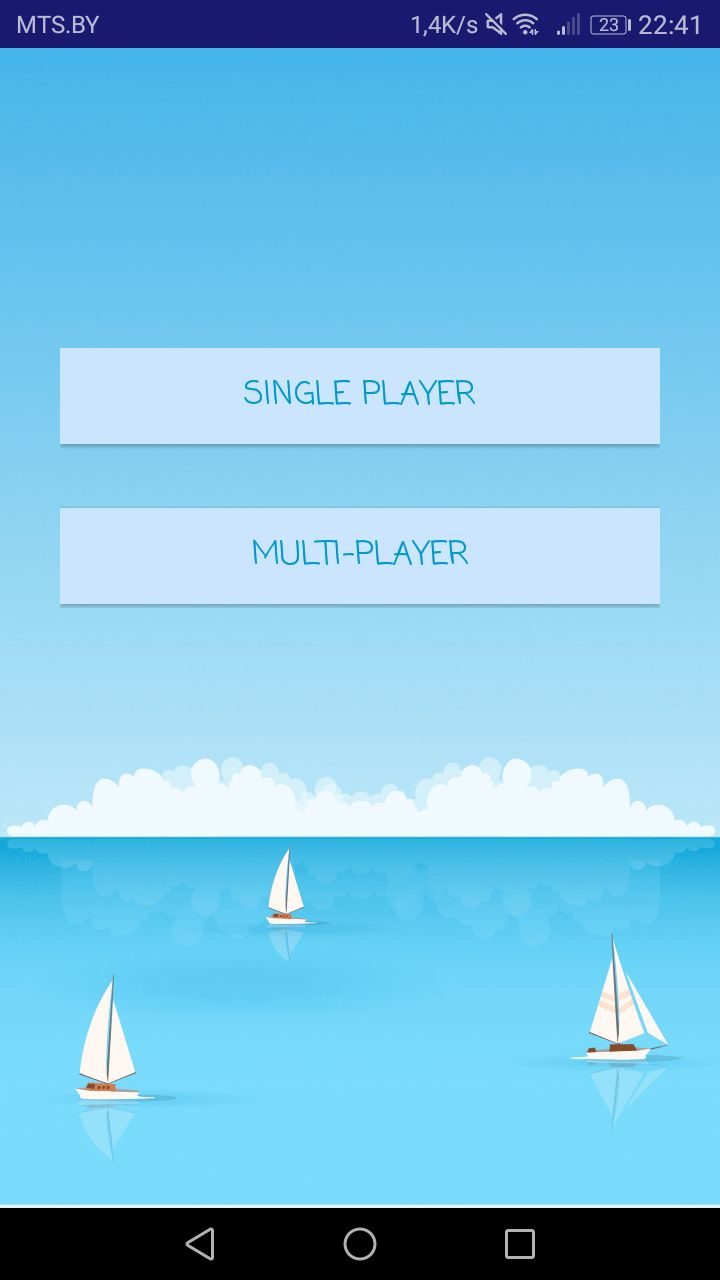
3.4 Тестирование ПО

Для тестирования приложения используем SmokeTest. SmokeTest (дымовое тестирование) в [тестировании программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) означает минимальный набор тестов на явные [ошибки](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B3). «Дымовой тест» обычно выполняется самим программистом; не проходившую этот тест программу не имеет смысла отдавать на более глубокое тестирование.

Список тестов, которые прошло клиент-серверное приложение, представлен ниже:

* 1. Запуск приложения.
  2. Переход в режим боя против бота.
  3. Переход в режим боя против реального противника.
  4. Возможность расставления кораблей.
  5. Возможность поворота кораблей.
  6. Возможность смены сложности бота.
  7. Возможность отправления запроса противнику.
  8. Использование игровых способностей.
  9. Возможность отключения звука.
  10. Выход в главное меню.
  11. Результат о победе.
  12. Результат о проигрыше.

Успешный запуск игрового приложения представлен на рисунке 3.2.

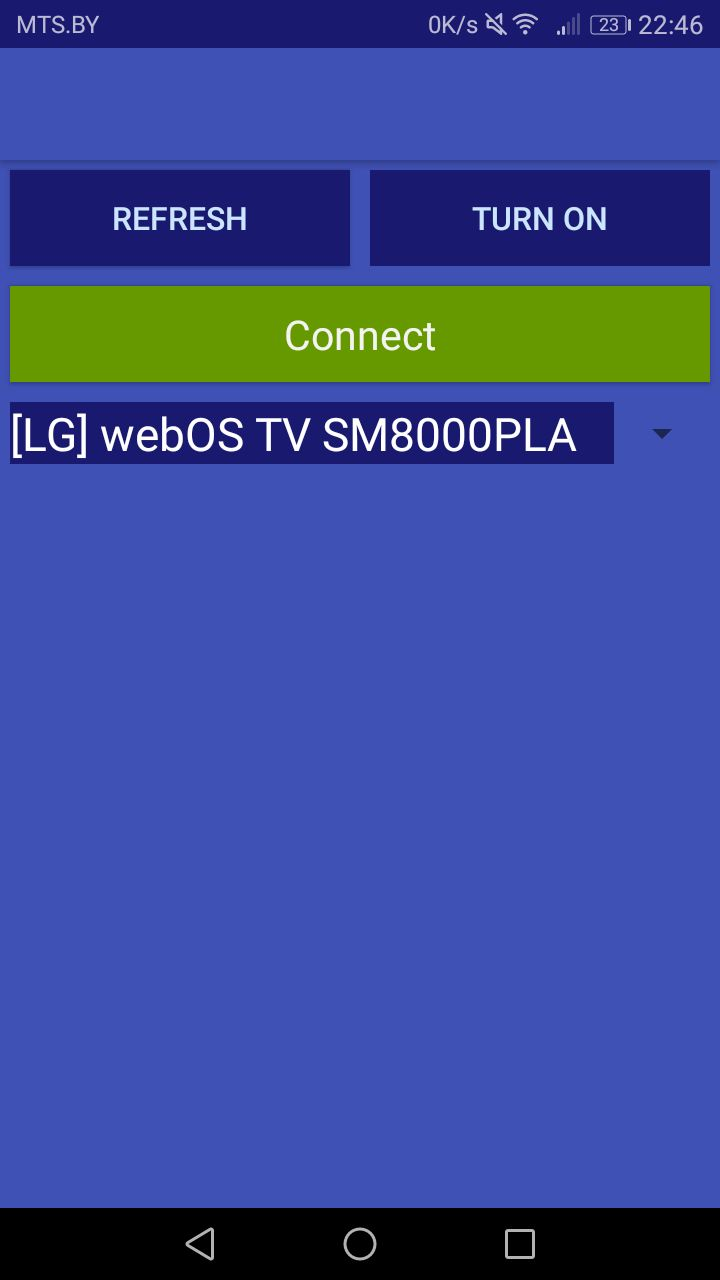


**Рисунок 3.2** – Успешный запуск игрового приложения

Далее была осуществлена проверка перехода во всевозможные режимы. Результат теста представлен на рисунке 3.3 и 3.4.

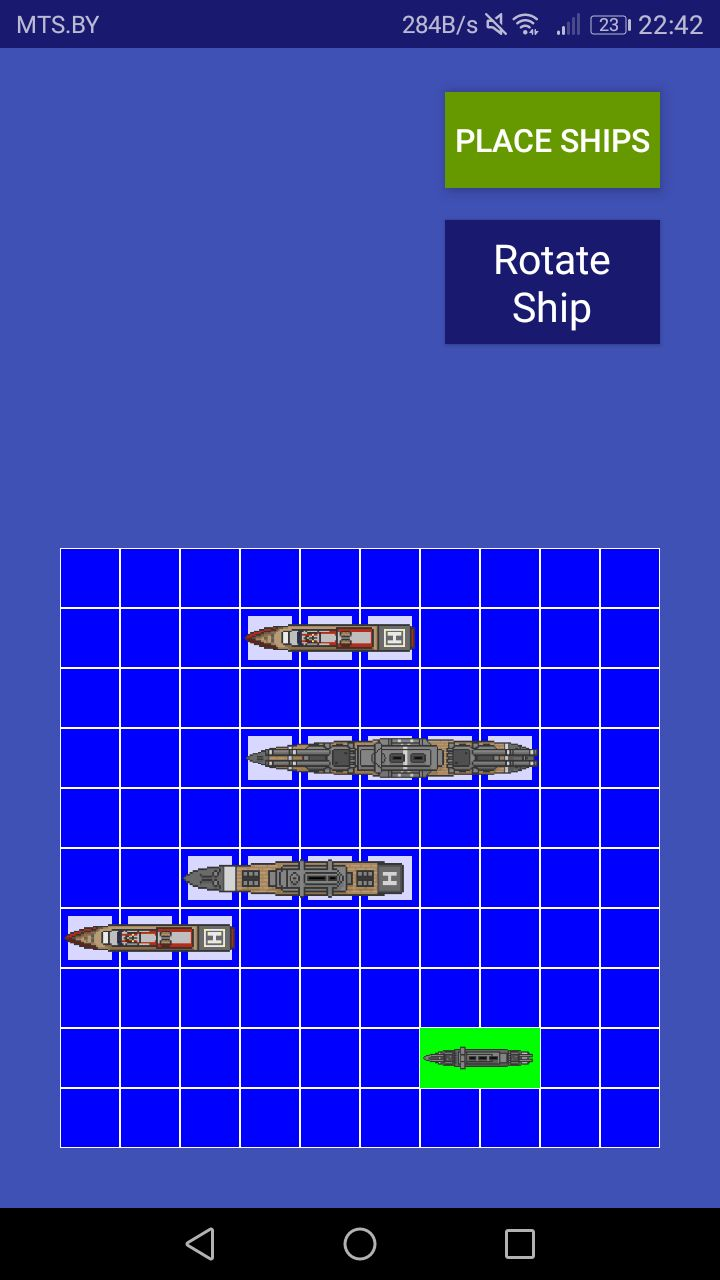


**Рисунок 3.3**– Режим против бота

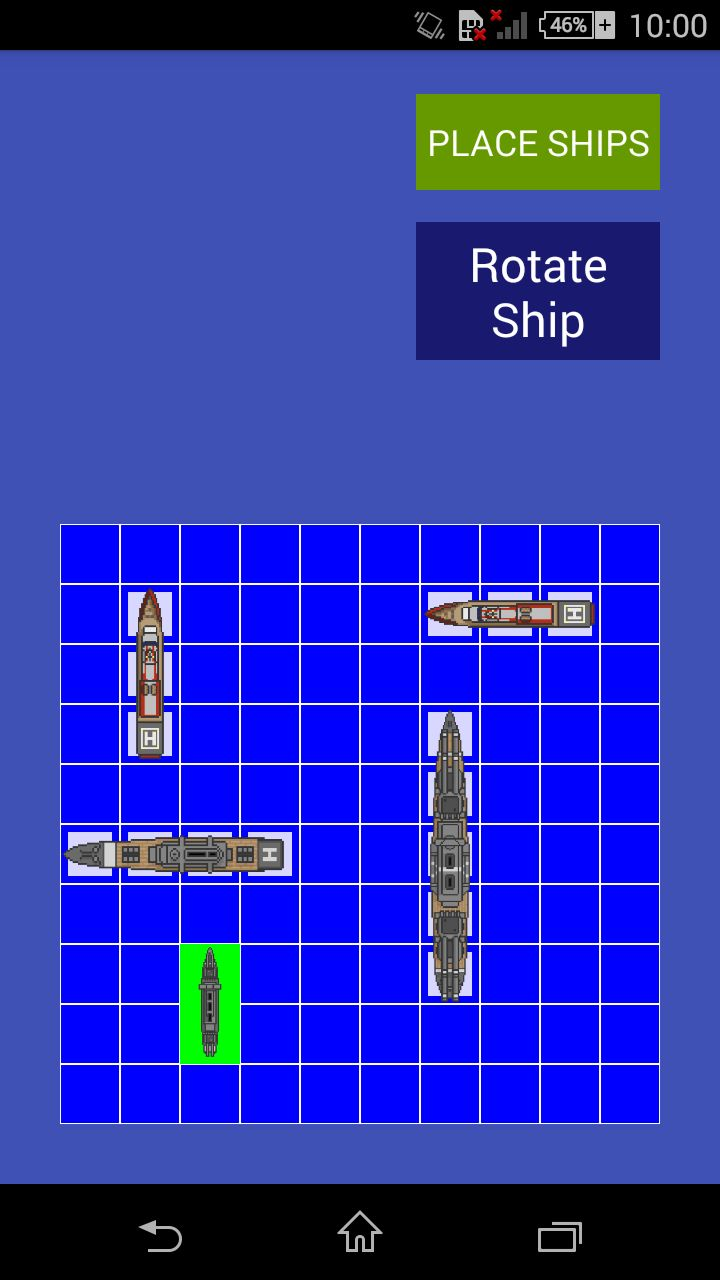


**Рисунок 3.4**– Режим против реального противника

Далее были проведены тесты на возможность расставления кораблей на поле боя. Все корабли успешно отображались и были доступны для игры, результаты представлены на рисунках 3.5 и 3.6.

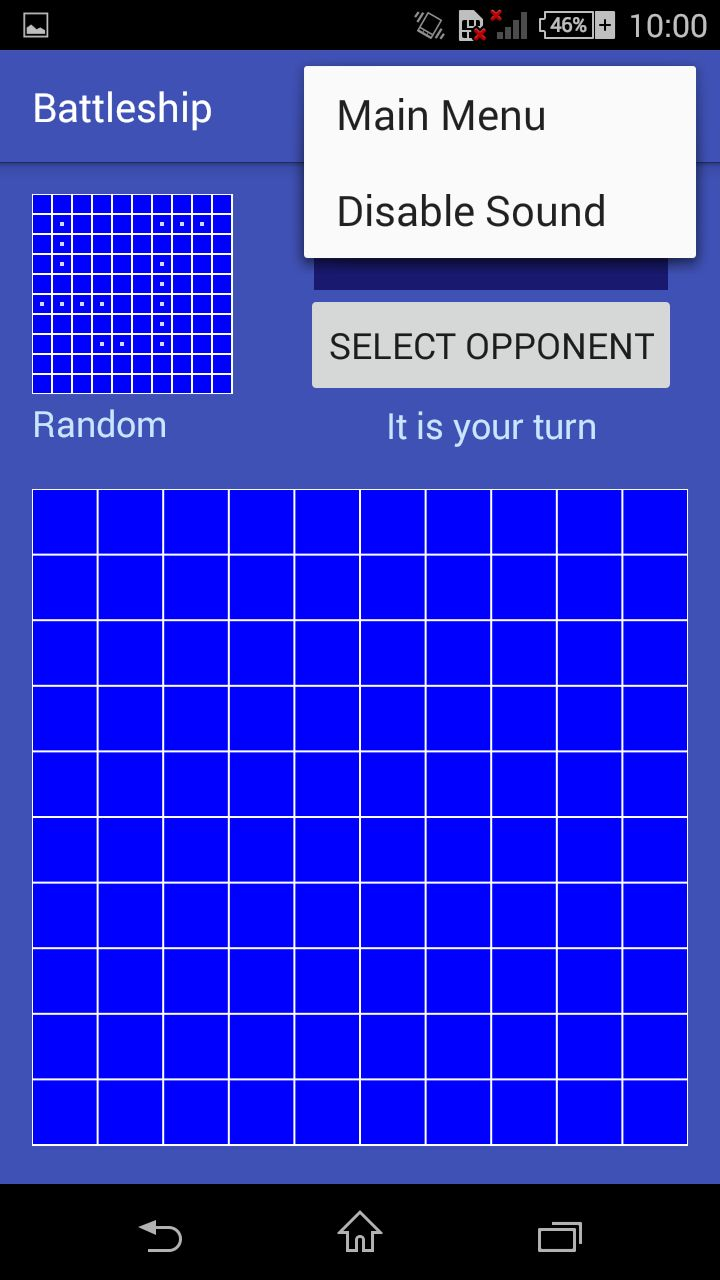


**Рисунок 3.5**– Расставление кораблей



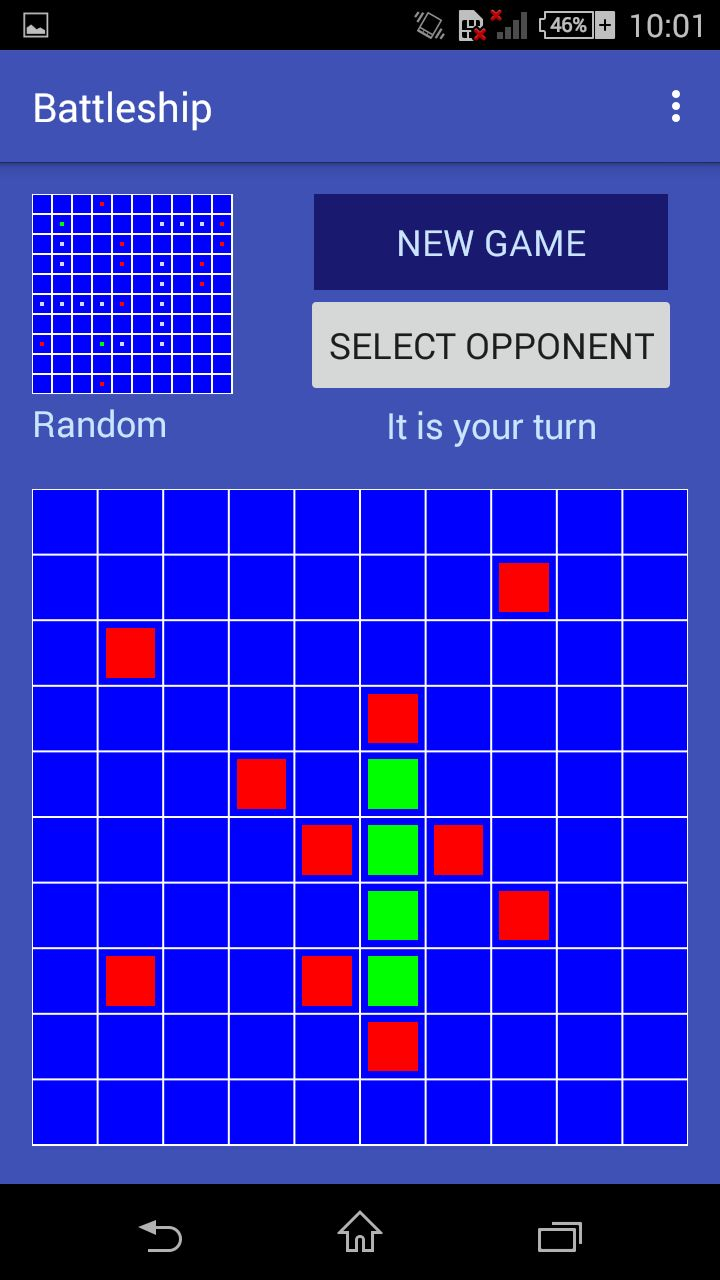
**Рисунок 3.6** – Поворот кораблей

Также был успешно пройден тест по смене сложности бота и отключения звука приложения. Результат теста представлен на рисунке 3.7 и 3.8.



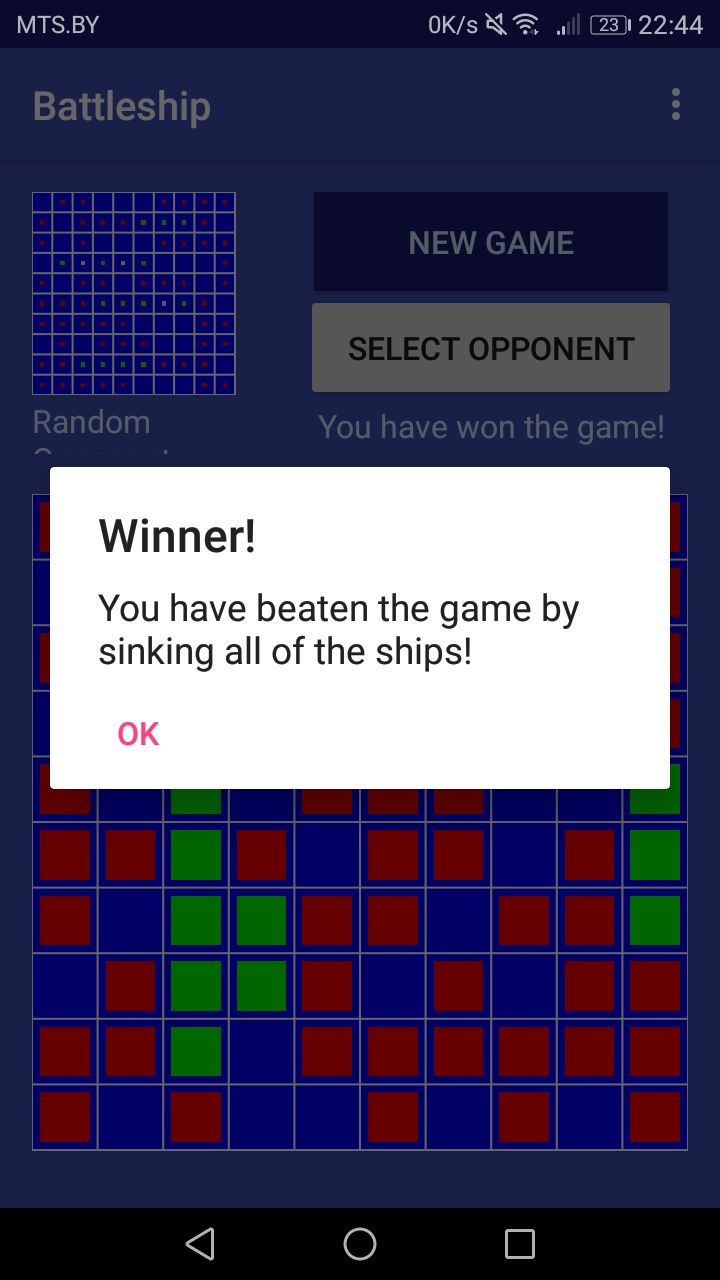
**Рисунок 3.8** – Отключение звука

Тест на проверку игровых способностей также был успешно пройден. Пользователю предоставляется полный контроль над игрой. Его результат представлен на рисунке 3.9

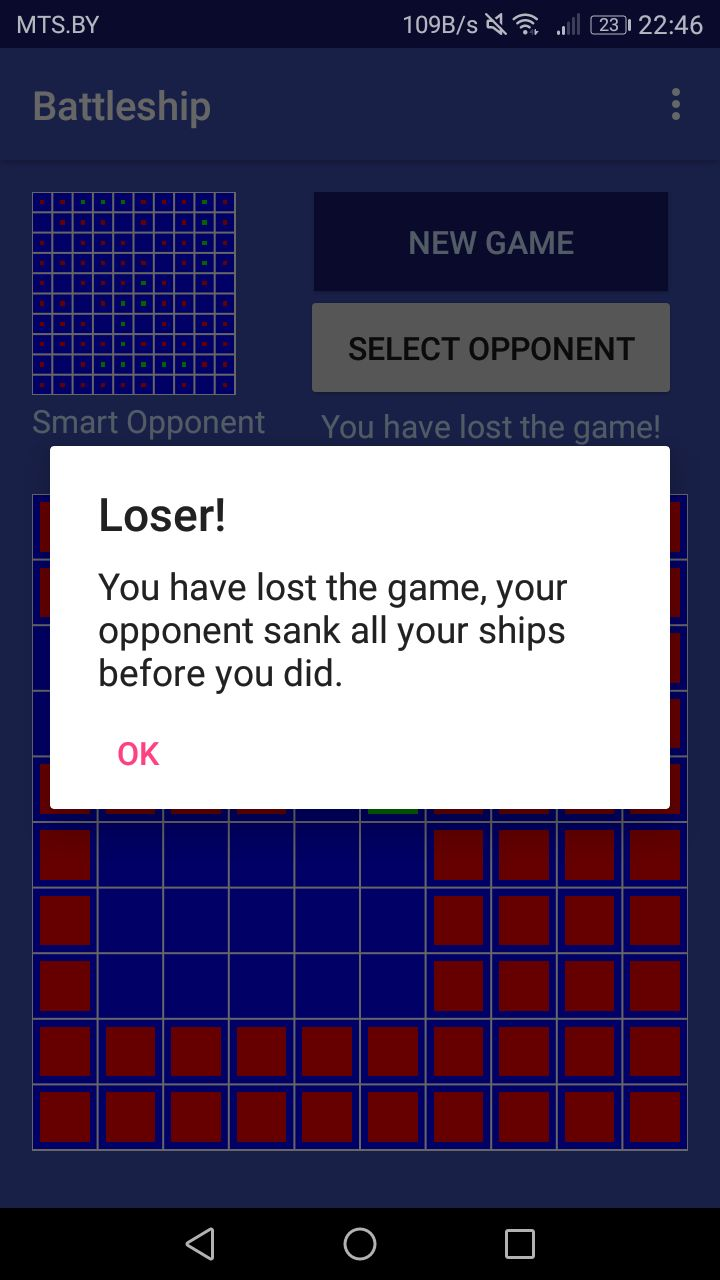


**Рисунок 3.9**– Проверка игровых способностей

Далее были успешно пройдены тесты на отображение победителя и проигравшего в боевом режиме. Результат данных тестов представлен на рисунке 3.10 и 3.11.



**Рисунок 3.10** – Результат о победе



**Рисунок 3.10** – Результат о проигрыше

По завершению тестирования можно сделать вывод, что программный продукт полностью готов к использованию, об этом свидетельствует успешное прохождение всех тестов.

ЗАКЛЮЧЕНИЕ

В результате выполнения данного курсового проекта, было сделано игровое мобильное приложение «Морской бой». Приложение написано на языке Java под операционную систему Android.

Мною были освоены навыки программирования на языке Java, а также навыки работы с платформой Android Studio. Были разработаны средства управления выстрелами, обработка событий при нажатии на кнопки, создание анимацией и было реализовано музыкальное сопровождение.

Были изучены и использованы встроенные компоненты Android Studio, отвечающие за столкновение с другими объектами, компоненты, отвечающие за реализацию анимации в игровом процессе, а также были использованы компоненты, прорисовку 2D изображений в игре.

В ходе разработки программы были придуманы алгоритмы для реализации вражеского бота, против которого вёлся бой. Также предоставлена возможность смены сложности бота.

В ходе проведения тестирования, ошибок обнаружено не было. Программа прошла все тесты с положительным результатом.

В конечном счёте, были соблюдены все критерии технического задания, игра работает исправно и без перебоев. Все технические проверки пройдены в соответствии со всеми положениями. На выходе получился окончательный продукт готовый к использованию широким кругом пользователей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. StackOverFlow[Электронный ресурс] – Режим доступа: <https://ru.stackoverflow.com/>. Дата обращения: 30.09.2020 – 2.12.2020.
2. Оформление дипломных работ по ГОСТу [Электронный ресурс] – Режим доступа: <http://einsteins.ru/oformlenie-diplomnoy> . Дата обращения: 01.10.2020.
3. Habr [Электронный ресурс] – Режим доступа: <https://habr.com/ru/post/175037/>. Дата обращения: 30.09.2020 – 2.12.2020.
4. Википедия [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org>. Дата обращения: 15.11.2020.
5. Начало разработки под Android[Электронный ресурс] – Режим доступа <https://tproger.ru/>how-to-start-android/. Дата обращения: 27.09.2020.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

Техническое задание

Введение

Наименование программного продукта – игровое приложение «Морской бой».

Эта программа предназначена для использования в развлекательных целях. Игра представляет собой соревновательное состязание между ботом и реальным противником.

А.1 Основание для разработки

Игра «Морской бой» разрабатывается в рамках курсового проекта студента учреждения образования «Полоцкий государственный университет» Яковлева Н.А. Основанием для разработки является выданное задание к курсовому проекту по теме разработки программы для развлекательных целей, утверждённое заведующим кафедры технологий программирования Петрович О.М. от 04.09.2020.

А.2 Назначение разработки

Функциональное и эксплуатационное назначение программы – это возможность скрасить свой досуг посредством использования данной программы.

А.3 Требования к программному продукту

А.3.1 Требования к функциональным характеристикам

При разработке программы «Морской бой» выдвинуты следующие требования к функциональным характеристикам:

1. Возможность выбора противника:

* игра против бота;
* игра против реального противника.

1. Возможность использовать все игровые способности;
2. Возможность изменения сложности бота;
3. Каждое действие должно сопровождаться музыкальным сопровождением;
4. Программа должна иметь интуитивно понятный интерфейс.

А.3.2 Требования к надежности

Данная программа должна надежно функционировать и обеспечивать сохранение игрового процесса. При возникновении аппаратного или программного сбоя программа должна оповещать пользователя о проблеме.

**А.3.3 Условия эксплуатации**

Эксплуатация программы «Морской бой» должна осуществляться на мобильном телефоне, на базе операционной системы Android. Минимальные требования к пользователю – умение обращаться с мобильным телефоном, знание основ работы с Android.

А.3.4 Требования к составу и параметрам технических средств

Для обеспечения устойчивости работы программного средства требуется:

* 1. Версия Android4.4 и выше.
  2. Процессор ARMv6, ARMv7.

А.3.5 Требования к информационной и программной   
совместимости

Программное средство должно удовлетворять следующему требованию: Android 4.4 и выше.

А.3.6 Требования к маркировке и упаковке

Требования к маркировке и упаковке отсутствуют.

А.3.7 Требования к транспортированию и хранению

Программное средство должно храниться на электронном носителе в виде исполняемого файла.

А.4 Требования к программной документации

Программная документация по приложению «Морской бой» должна быть предоставлена в следующем составе:

1. техническое задание. Согласно ГОСТ 19.201-78;
2. пояснительная записка. Согласно ГОСТ 19.101-77.

Требования к перечисленным программным документам устанавливаются государственными стандартами ЕСПД.

А.5 Стадии и этапы разработки

Разработка программы заключается в следующем:

1. Анализ исходных данных и постановка задачи проектирования, разработка технического задания.
2. Разработка интерфейса, архитектуры и структуры программы.
3. Реализация и тестирование программы.
4. Разработка программной документации.

**А.6 Порядок контроля и приемки**

Контроль и приемка программного средства осуществляется в соответствии с программой и методикой испытаний.

Для проверки корректности приложения применялись следующие программные средства:

1. ОС Linux Manjaro;
2. среда разработки Android Studio.

Тестирование программы состояло из проверки корректности работы ранее перечисленных функций.

Методы испытаний:

Основным методом испытания программы является визуальный контроль выполнения программой требующихся функций, корректное выполнение юнит-тестов.

ПРИЛОЖЕНИЕ Б

**Диаграмма вариантов использования**

