Sprawozdanie Projektowe

Przedmiot:

- ***Programowanie w języku Java (projekt)***

Temat:

- ***Gra Statki***

Zespół:

- ***Dominik Grudzień***
- ***Patryk Grzywacz***
- ***Adrian Pełka***

Informacje:

- ***Rok: 2***
- ***Kierunek: Informatyka***
- ***Grupa: 2ID12A***

Szczegółowy Opis Projektu

Język:

- **Obiektowa JavaSE-15**

Technologia:

- **Silnik LIBGDX w wersji 1.9.14 na podstawie frameworka LWJGL**
- **Framework JUNIT5 w wersji 5.6.0 do testu jednostkowego**
- **GRADLE Project Manager w wersji 6.7.1**
- **Do tworzenia użyto IDE: VSCODE z rozszerzeniami umożliwiającymi pisanie projektów w Javie, kontrolowanie zdalnym repozytorium GitHub i innymi kluczowymi technologiami dla projektu .**

Biblioteki:

- **Standardowe biblioteki języka Java wersji 15**
- **Biblioteki LIBGDX – do tworzenia aplikacji desktopowej wraz z całą zawartością.**
- **Biblioteki Junit Jupiter oraz Junit Platform – do przeprowadzenia testu jednostkowego.**
- **Własne napisane biblioteki jak np. do prowadzenia rozgrywki, kontrolowania AI przeciwnika czy też manipulowania obiektami gry.**

Funkcjonalności:

- **Możliwość samodzielnego rozmieszczenia na planszy statków różnych wielkości i ich obrotu, lub też skorzystanie z przycisku do automatyzacji tych działań .**
- **Rozgrywanie gry w statki wersji papierowej wg. klasycznych zasad przeniesione w system graficzny 2D.**
- **Prowadzenie bitwy w otoczeniu specjalnych efektów graficznych i audio na żywo z wymagającym przeciwnikiem komputerowym.**
- **Bonusy w postaci Combo ,które zwiększa w czasie trwania uzyskiwane punkty za trafienia i zniszczenia.**
- **Możliwość zapisu i odczytu wyników najlepszych graczy oraz ustawień z plików .txt.**
- **Interaktywne menu gry za pomocą myszki.**
- **Klimatyczna muzyka.**

Uruchamianie oraz obsługa projektu:

- **Na systemie Windows:**
  - **Proste uruchomienie pliku ShipsGameV_1.2.exe**
  - **Lub poprzez uruchomienie ShipsDist.jar poprzez terminal komendą java -jar ShipsDist.jar**

    - Ewentualnie można też stworzyć projekt w dowolnym IDE poprzez Gradle i przekopiować cały folder projektu z plikami źródłowymi, assetami , ustawieniami Gradle itd. do właściwego folderu nowego projektu i skompilowanie go.
- Teoretycznie na innych systemach typu Linux:
    - Poprzez uruchomienie ShipsDist.jar poprzez terminal komendą java -jar ShipsDist.jar

Informacje na temat klas oraz metod:

- Klasa DesktopLauncher – Główna klasa programu.
    - Public static void main(String[]) – Główna metoda odpowiedzialna za połączenie silnika z projektem Java i utworzenie aplikacji.
- Klasa Main – Główna klasa operująca oknami gry.
    - Public void create() – Metoda odpowiedzialna za stworzenie okna gry i ustawienie ekranu na Menu Główne.
    - Public void dispose() – Metoda odpowiedzialna za niszczenie elementów silnika libgdx.
- Klasa MenuScreen – Klasa zawierająca główne menu gry.
    - Public MenuScreen(Main) – Konstruktor obiektu klasy MenuScreen.
    - Private void update(float) – Metoda odpowiedzialna za odświeżanie operacji i wyglądu w menu gry.
    - Public void render(float) – Metoda odpowiedzialna za renderowanie okna menu gry.
    - Public void show() – Metoda odpowiedzialna za tworzenie, ustawianie i ładowanie elementów w głównym menu gry.
    - Public void pause() – Metoda obsługująca pauzę aktywności okna w menu.

    - Public void resume() – Metoda obsługująca wznowienie aktywności okna w menu.
    - Public void resize(int , int) – Metoda obsługująca zmianę rozmiaru okna w menu.
    - Public void hide() - Metoda obsługująca ukrycie okna w menu.
    - Public void dispose() - Metoda obsługująca niszczenie elementów silnika libgdx oraz menu głównego.
- Klasa MenuGlobalElements – Klasa zawierająca metody i pola wykorzystywane przez okna menu.
    - Public MenuGlobalElements(Game) – Konstruktor obiektu klasy tworzący assety menu i ustawiający inne atrybuty menu.
    - Public void moveMenu(float) – Metoda odpowiedzialna za animację tła w menu gry.
    - Public void disposeMenu() – Metoda odpowiedzialna za niszczenie elementów wykorzystywanych przez menu.
- Klasa HelpScreen – Klasa okna pomocy w głównym menu.
    - Public HelpScreen(Main) – Konstruktor obiektu okna.
    - Private void update(float) – Metoda odpowiedzialna za odświeżanie operacji i wyglądu w oknie pomocy menu.
    - Public void render(float) – Metoda odpowiedzialna za renderowanie elementów okna pomocy.
    - Public void show() – Metoda odpowiedzialna za tworzenie, ustawianie i ładowanie elementów w oknie pomocy menu.
    - Public void pause() – Metoda obsługująca pauzę aktywności okna w oknie pomocy menu.
    - Public void resume() – Metoda obsługująca wznowienie aktywności okna w oknie pomocy menu.
    - Public void resize(int , int) – Metoda obsługująca zmianę rozmiaru okna w oknie pomocy menu.

    - Public void hide() - Metoda obsługująca ukrycie okna w oknie pomocy menu.
    - Public void dispose() - Metoda obsługująca niszczenie elementów silnika libgdx oraz okna pomocy menu.
- Klasa OptionScreen – Klasa opcji w menu gry.
    - Public OptionScreen(Main) – Konstruktor obiektu okna.
    - Private void update(float) – Metoda odpowiedzialna za odświeżanie operacji i wyglądu w oknie opcji menu.
    - Public void render(float) – Metoda odpowiedzialna za renderowanie elementów okna opcji.
    - Public void show() – Metoda odpowiedzialna za tworzenie, ustawianie i ładowanie elementów w oknie opcji menu.
    - Public void pause() – Metoda obsługująca pauzę aktywności okna w oknie opcji menu.
    - Public void resume() – Metoda obsługująca wznowienie aktywności okna w oknie opcji menu.
    - Public void resize(int , int) – Metoda obsługująca zmianę rozmiaru okna w oknie opcji menu.
    - Public void hide() - Metoda obsługująca ukrycie okna w oknie opcji menu.
    - Public void dispose() - Metoda obsługująca niszczenie elementów silnika libgdx oraz okna opcji menu.
    - Private void saveSettings() – Metoda odpowiedzialna za zapis ustawień z okna opcji do pliku tekstowego settings.txt wykorzystywanego w całej aplikacji.
- Klasa ScoreScreen – Klasa okna wyników w menu.
    - Public ScoreScreen(Main) – Konstruktor obiektu okna oraz obiektu klasy Scores do wyświetlania wyników.
    - Private void update(float) – Metoda odpowiedzialna za odświeżanie operacji i wyglądu w oknie wyników menu.
    - Public void render(float) – Metoda odpowiedzialna za renderowanie elementów okna wyników.

- Public void show() – Metoda odpowiedzialna za tworzenie, ustawianie i ładowanie elementów w oknie wyników menu.
- Public void pause() – Metoda obsługująca pauzę aktywności okna w oknie wyników menu.
- Public void resume() – Metoda obsługująca wznowienie aktywności okna w oknie wyników menu.
- Public void resize(int , int) – Metoda obsługująca zmianę rozmiaru okna w oknie wyników menu.
- Public void hide() - Metoda obsługująca ukrycie okna w oknie wyników menu.
- Public void dispose() - Metoda obsługująca niszczenie elementów silnika libgdx oraz okna wyników menu.

- Klasa Scores – Klasa, której rolą jest tworzenie pliku scores.txt oraz przechowywanie wszystkich wyników.
  - Klasa protected Node – Klasa zagnieżdżona przechowująca jeden rekord wyników
    - Protected Node(String , float , float , float) – Konstruktor obiektu rekordu z danymi przekazanymi mu przez parametr.
  - Klasa protected SortByScore – Klasa zagnieżdżona pozwalająca przeciążyć komparator do listy typu ArrayList.
    - Public int compare(Node , Node) – Przeciążona metoda z Comparatora ,która pozwala sortować klasie Scores rekordy wg. wyników.
  - Public Scores() – Konstruktor obiektu klasy Scores, tworzący listę typu ArrayList i wypełniający ją rekordami.
  - Private void loadScores() – Metoda ta tworzy plik z wynikami jeśli nie istnieje lub pobiera dane z pliku i umieszcza je w rekordach listy a następnie sortuje listę.

    - Public void drawScores(SpriteBatch , BitmapFont , float , float) – Metoda odpowiedzialna za rysowanie rekordów z listy na ekranie.
- Klasa Animator – Klasa animująca sprite'y w grze.
    - Public Animator(Texture , Vector2 , float) – Konstruktor główny obiektu klasy
    - Public void setStartAnimation() – Metoda resetująca animacje
    - Public void update() – Metoda aktualizująca animacje.
    - Public void update(int) – Metoda aktualizująca animacje w określonym wierszu SpriteSheet.
    - Public TextureRegion getCurrentFrame() – Metoda zwracająca aktualną klatkę animacji.
    - Public TextureRegion getIdleAnimation() – Metoda zwracająca animację spoczynku.
- Klasa BoomEffect – Klasa odpowiedzialna za tworzenie efektów trafień lub nietrafień oraz zniszczeń.
    - Public BoomEffect(Sound , Texture) – Konstruktor tworzący efekt trafienia.
    - Public BoomEffect(Sound , Texture , Vector2 , float) – Konstruktor tworzący efekt nietrafienia.
    - Public BoomEffect(Sound , Texture , boolean) – Konstruktor tworzący efekt zniszczenia.
    - Public Vector2 getPos() – Metoda zwracająca pozycję efektu.
    - Public void setPos(Vector2f) – Metoda ustawiająca pozycję efektu.
    - Public void setPos(Vector2f , int , int) – Metoda tworząca i ustawiająca wiele efektów zniszczeń.
    - Public void updateAnimation() – Metoda do aktualizacji animacji efektu.
    - Public void updateAnimation(boolean) – Metoda do aktualizacji animacji efektów znisczeń.

- Public void drawEffect(SpriteBatch) – Metoda do rysowania efektu na ekranie.
- Public void drawEffect(SpriteBatch , boolean) – Metoda do rysowania efektów zniszczeń na ekranie.
- Public boolean playSound(float) – Metoda do włączania dźwięku efektu.
- Public void playSound(boolean , float) – Metoda do włączania dźwięku efektów zniszczeń.
- Public void resetAnimation() – Metoda do resetowania animacji efektu/efektów do stanu początkowego.

- Klasa ShootParticleEffect – Klasa przechowująca wszystkie efekty strzałów pojedynczego statku.
  - Public ShootParticleEffect(Texture , float , float , Vector2 , int) – Konstruktor obiektu
  - Public void setPositions(GameObject) – Zadaniem tej metody jest poprawne ulokowanie efektów w zależności od pozycji i rotacji statku i wieżyczek.
  - Public void updateAnimation(GameObject) – Metoda do aktualizowania animacji wszystkich efektów.
  - Public void drawAnimation(SpriteBatch) – Metoda do rysowania efektów na ekranie.
  - Public void resetAnimation() – Metoda do resetowania animacji do stanu początkowego.
- Interface Constant – Interfejs przechowujący kilka stałych do obliczeń i ustawiania logiki gry.
- Klasa Score – Klasa przechowująca wyniki gracza i komputera w czasie bitwy
  - Public Score(int) – Konstruktor obiektu klasy Score nadający identyfikator .
  - Public void setPlayerName(String) – Metoda nadająca nazwę do obiektu.
  - Public void drawInfo(BitmapFont , SpriteBatch , float , float, int , int , int , Texture[]) – Metoda do

rysowania na ekranie elementów informacyjnych z obiektu.
    - Public void update(int[][]) – Metoda do aktualizacji danych odnośnie strzałów.
    - Public void updateTime(float) – Metoda aktualizująca czas trwania tur właściciela obiektu.
    - Public void addPointsForHit() – Metoda przyznająca punkty za trafienie.
    - Public void addPointsForDestroy(int) – Metoda przyznająca punkty za zniszczenie statku.
    - Public void increaseCombo() – Metoda zwiększająca combo za każde trafienie pod rząd.
    - Public void zeroCombo() – Metoda zerująca combo.
    - Public float getScoreValue() – Metoda zwracająca wynik.
    - Public String getPlayerName() – Metoda zwracająca nazwę właściciela obiektu.
    - Public float getTimeElapsed() – Metoda zwracająca czas trwania tur.
    - Public float getAccuracyRatio() – Metoda zwracająca celność strzałów.
    - Public int getCombo() – Metoda zwracająca combo.
    - Public int getShipsDestroyed() - Metoda zwracająca ilość zniszczonych statków.
    - Public int getShotsMissed() – Metoda zwracająca ilość chybionych strzałów.
    - Public int getIdNumber() – Metoda zwracająca identyfikator.
- Klasa GameImageButton – Klasa reprezentująca przycisk obrazkowy
    - Public GameImageButton(float , float , Hud , Sprite[]) – Konstruktor obiektu klasy GameImageButton

    - Public GameImageButton(Sprite[]) – Drugi konstruktor obiektu klasy.
    - Public void setOptionsListener() – Metoda ustawiająca słuchacza do przycisku.
    - Public boolean getGameMenuState() – Metoda zwraca obecny stan okna.
- Klasa GameTextButton – Klasa reprezentująca przycisk tekstowy
    - Public GameTextButton(String , Skin , final int) – Konstruktor obiektu klasy używany podczas rozgrywki.
    - Public GameTextButton(String , float , float , Skin , final int , final Main) – Drugi konstruktor obiektu używany w głównym menu aplikacji.
    - Private void menuOptions(int) – Metoda przełącza ekran aplikacji na podstawie parametru.
- Klasa OptionsWindow – Klasa reprezenntująca okno dialogowe.
    - Public OptionsWindows(String , Hud) – Główny konstruktor obiektu.
    - Private void saveSettings() – Metoda zapisująca ustawienia gry do pliku.
    - Protected void result(final Object) – Metoda wynikowa po wciśnięciu przycisku w oknie dialogowym.
- Klasa Hud – Klasa Hud'u (heads-up display) gry.
    - Public Hud(AssetManager , Main , GameScreen , Cursor) – Konstruktor obiektu klasy.
    - Private void setButtonSprites(Texture[] , Sprite[] , float) – Metoda ustawiająca sprite'y przycisków.
    - Public void update() – Metoda aktualizująca elementy hud'u w czasie gry.
    - Public void dispose() – Metoda niszcząca elementy hud'u.

- Public GameImageButton getRepeatButton() – Metoda zwracająca przycisk losowego generowania pozycji statków.
- Public GameImageButton getPlayButton() – Metoda zwracająca przycisk przejścia do rozgrywki.
- Public String getPlayersName() – Metoda zwracająca nazwę gracza.
- Public Sprite getPlayButtonGreenStyle() – Metoda zwracająca Sprite przycisku Play.
- Public Dialog getPlayersSetNameDialog() – Metoda zwracająca okno dialogowe wpisywania nazwy gracza.
- Public Stage getStage() – Metoda zwracająca scenę elementów hud'u.
- Public Skin getSkin() – Metoda zwracająca styl skórki elementów hud'u.
- Public boolean isPasued() – Metoda zwracająca stan opcji podczas rozgrywki.

- Klasa ComputerPlayerAi – Klasa odpowiadająca za podejmowanie decyzji przez przeciwnika komputerowego.
  - Public ComputerPlayerAi(int[][]) – Konstruktor główny obiektu klasy.
  - Public float getX() – Metoda zwracająca pozycję x na osiX strzału.
  - Public float getY() – Metoda zwracająca pozycję y na osiY strzału.
  - Public void update(boolean , boolean , boolean , int[][] , GameObject[] , Vector2 , int) – Metoda do aktualizowania danych i logiki przeciwnika komputerowego.
  - Public boolean attackEnemy(float) – Metoda sygnalizująca przeprowadzenie ataku.
  - Private void Missed() – Metoda do obliczeń logiki i pozycji strzału podczas, gdy poprzedni strzał chybił.

    - Private void HittedNdestroyed() – Metoda do obliczeń logiki i pozycji strzału ,gdy poprzedni strzał zniszczył statek wroga.
    - Private void HittedAndNotDestroyed(boolean) – Metoda do obliczeń logiki i pozycji strzału ,gdy poprzedni strzał był chybiony a pozostały jeszcze trafienia nierozliczone.
    - Private int findNextSpot(int) – Metoda do odnalezienia poprawnej pozycji strzału przy uwzględnieniu wielu danych i logiki.
- Klasa ComputerPlayerAiTest – Klasa testu jednostkowego Junit
    - Public void test() – Metoda testująca poprawność obliczeń na podstawie zawartych już danych.
- Klasa GameSettings – Klasa zawierająca ustawienia gry.
    - Public GameSettings(Game) – Konstruktor obiektu klasy.
    - Private void loadSettings() – Metoda wczytująca ustawienia z pliku.
    - Public void playSound() – Metoda włączająca dźwięk kliknięcia.
    - Public void dispose() – Metoda niszcząca wszystkie elementy obiektu.
- Klasa GameSlider – Klasa reprezentująca suwak z libgdx
    - Public GameSlider(float , float , float , float , float , boolean , Skin , final Main) – Konstruktor obiektu klasy.
    - Public GameSlider(float , float , float , boolean , Skin) – Drugi konstruktor obiektu klasy.
    - Public void setSliderType(int , final GameSettings) – Metoda ustawiająca słuchacza suwaka.
    - Public void setSliderType(int) – Metoda ustawiająca słuchacza suwaka.

- Klasa GameObject – Klasa przechowująca wszystkie dane oraz metody dotyczące jednego statku.
  - Public GameObject(String , float , float) – Konstruktor obiektu z samą teksturą.
  - Public GameObject(Texture , float , float , boolean , boolean , Vector2) – Konstruktor obiektu z teksturą , sprite'm i animatorem do animacji sprite'a.
  - Public GameObject(Texture , Texture , float , float , boolean , int , Vector2) – Konstruktor obiektu z dwoma teksturami i dwoma sprite'ami , po jednym dla statku i jego fal.
  - Public GameObject(Texture , Texture , Texture[] , float , float , boolean , int , Vector2) – Konstruktor obiektu z dwoma teksturami i sprite'ami , po jednym dla statku i jego fal oraz z sprite'ami jego wieżyczek.
  - Public int getShipSize() – Metoda do zwracania wielkości statku
  - Public float getTurretRotation(int) – Metoda zwracająca obrót w stopniach pojedynczej wieżyczek określanej parametrem.
  - Public Sprite[] getTurrets() – Metoda do zwracania tablicy sprite'ów wieżyczek.
  - Public void updateAnimation() – Metoda do aktualizacji animacji statku.
  - Public void updateTexture() – Metoda do aktualizacji animacji fal statku.
  - Public void moveTexture(float) – Metoda do przesuwania tekstury wg. osiX.
  - Protected void createSprite(Texture) – Metoda do tworzenia sprite'a obiektu oraz innych potrzebnych zasobów.

- Protected void createSprite(Texture , int) – Metoda do tworzenia sprite'a statku i innych jego elementów w oparciu o jego rozmiar.
- Protected void createTurrets(Texture[]) – Metoda do tworzenia tablicy sprite'ów wieżyczek statku oraz ustawienia ich na poprawnej pozycji.
- Public void drawTurrets(SpriteBatch) – Metoda do rysowania sprite'ów wieżyczek statku na ekranie.
- Public void drawTurrets(SpriteBatch , boolean) – Metoda do rysowania sprite'ów wieżyczek statku wroga na ekranie.
- Public Texture getTexture() – Metoda zwracająca teksturę statku.
- Public Sprite getSprite() – Metoda zwracająca sprite'a statku.
- Public int getRotation() – Metoda do zwracania kierunku obrotu statku.
- Protected void createSpriteWave(Texture) – Metoda do tworzenia sprite'a fal statku.
- Public Texture drawTexture() – Metoda do zwracania tekstury do rysowania.
- Public void drawSprite(SpriteBatch , boolean) – Metoda do rysowania statku wroga i jego fal.
- Public void drawSprite(SpriteBatch) – Metoda do rysowania statku i jego fal.
- Public void drawSprite(SpriteBatch , boolean , boolean , ShapeRenderer , boolean) – Metoda do rysowania statku wroga i jego fal oraz prostokąta informującego.
- Public void drawSprite(SpriteBatch , boolean , boolean , ShaperRenderer) – Metoda do rysowania statku i jego fal oraz prostokąta informującego.

- Public void setSpritePos(Vector2) – Metoda do ustawiania nowej pozycji statku i jego wszystkich elementów.
- Public void translate(Vector2) – Metoda do przesuwania statku wg. osi X oraz osi Y.
- Public void translateX(float) – Metoda do przesuwania statku wg. osi X.
- Public void translateY(float) - Metoda do przesuwania statku wg. osi Y.
- Public boolean spriteContains(Vector2) – Metoda do sprawdzania czy punkt z parametru znajduje się w spri'cie
- Public void changeRectColour() – Metoda do określenia koloru prostokąta w zależności od jego ustawienia i kolizji.
- Public void setGoodPlacement(boolean) – Metoda do zmiany wartości określającej czy statek jest w dobrej pozycji.
- Public void destroyElement() – Metoda do niszczenia pojedynczego elementu statku.
- Public void changeDestroyTexture(Texture , Texture[]) – Metoda do zmiany tekstur statku na zniszczone tekstury.
- Public void checkDestroyment() – Metoda określająca czy statek został w całości zniszczony.
- Public boolean isDestroyed() – Metoda zwracająca określenie czy statek jest zniszczony.
- Public boolean collide(Rectangle) – Metoda do sprawdzania kolizji z innym statkiem na planszy.
- Public boolean collide(Rectangle , boolean , boolean) – Metoda do sprawdzania kolizji z innym statkiem na planszy , który ma inne obrócenie.

    - Public void rotate90() – Metoda obracająca cały statek wraz z jego elementami o 90 stopni w prawo.
    - Public void placeTurretsAccordingly() – Metoda do ustawienia wieżyczek w odpowiednim miejscu na statku.
    - Public void rotateTurret(float , int) – Metoda ustawiająca daną wieżyczkę na dany kąt obrotu.
    - Public Vector2f getVectorPos(int) – Metoda zwracająca pozycję danej wieżyczki.
    - Public Vector2f getPosition() – Metoda do zwracania pozycji statku.
- Klasa GameEngine – Klasa abstrakcyjna zawierająca klasy , metody oraz obiekty i inne zmienne niezbędne do funkcjonowania aplikacji i rozgrywki.
    - Klasa Board – Klasa zagnieżdzona służąca do reprezentacji planszy oraz obliczeń logiki do rozgrywki.
        - Protected Board(int , int) – Konstruktor obiektu klasy.
        - Protected void placeShipOnBoard(int) – Metoda wypełniająca tablice logiczne do rozgrywki na podstawie pozycji statków.
        - Protected void hitShip(int , int) – Metoda sprawdzająca , który z statków tej planszy został trafiony lub zniszczony oraz utworzenie animacji trafienia lub zniszczenia statku.
    - Protected void switchTurn() – Metoda do zmiany tury gry.
    - Protected void loadGameEngine(AssetManager) – Metoda do ładowania assetów gry do AssetManagera.
    - Protected void loadHudAssets(AssetManager) – Metoda do ładowania assetów interfejsu do AssetManagera.
    - Protected boolean preparation(boolean , AssetManager) – Metoda do utworzenia obiektów oraz zmiennych do działania gry.

- Protected void generateAndPlaceShipsOnBoard(int , boolean) – Metoda do automatycznego rozmieszczenia statków i ustawienia logiki planszy na której się znajdują.
- Protected void touchDownSprite(int , int) – Metoda do określenia na ,który statek kliknięto i przytrzymany lewy klawisz myszki do Drag&Drop.
- Protected void touchUpSprite() – Metoda do aktualizacji logiki o rozmieszczeniu statków.
- Protected void dragSprite(int , int) – Metoda do poruszania statków w czasie rozmieszczania na polu bitwy metodą Drag&Drop.
- Protected boolean isShipPlacedGood(GameObject , int) – Metoda do sprawdzania czy statek znajduje się w dopuszczalnej pozycji na planszy.
- Protected void rotateActualShip() – Metoda do obracania statku aktualnie trzymany przy Drag&Drop po wciśnięciu klawisza R.
- Protected void drawStage2Text(BitmapFont , SpriteBatch) – Metoda do rysowania tekstu pomocniczego na ekranie w czasie przed bitwą.
- Protected boolean checkAllShips() – Metoda sprawdzająca czy wszystkie statki są na dobrych pozycjach przed bitwą.
- Protected void rotateTurretsWithMouse(float , float) – Metoda do obracania wieżyczkami statków podczas bitwy i własnej tury.
- Protected void checkHit(int , int) – Metoda do sprawdzania czy trafiono w któryś okręt na planszach.
- Protected boolean shoot(int , int) – Metoda do oddawania strzałów w określoną pozycję.

    - Protected void checkEnemyBoard(int , int) – Metoda do sprawdzania czy gracz może oddać strzał w danej chwili na daną pozycję.
    - Protected void dispose() – Metoda do zwalniania zasobów gry tworzonych przez tę klasę.
- Klasa GameScreen – Klasa ekranu głównego gry , gdzie znajduje się rozgrywka.
    - Public GameScreen() – Konstruktor ekranu głównego gry.
    - Private void drawMap() – Metoda do renderowania mapy.
    - Private void drawShipsEnTurrets() – Metoda do renderowania statków.
    - Private void drawHit(float) – Metoda do renderowania efektu trafienia.
    - Private void drawMiss(float) – Metoda do renderowania efektu chybienia.
    - Private void drawDestroyment(float) – Metoda do renderowania efektu zniszczenia statku.
    - Private void drawShootingEffect(float) – Metoda do renderowania efektów strzałów okrętów.
    - Private void drawMarks(SpriteBatch) – Metoda do renderowania efektów informacyjnych dla gracza.
    - Private void createDialog() – Metoda do stworzenia okna dialogowego po skończonej bitwie.
    - Private void drawScores(SpriteBatch) – Metoda do renderowania informacji o wynikach gracza i komputera.
    - Private void drawLoadingScreen() – Metoda do renderowania ekranu ładowania.
    - Private void drawExitScreen() – Metoda do renderowania wiadomości po skończonej bitwie.

- Private void createMap() – Metoda do utworzenia mapy z zasobów z AssetManagera.
- Private void createFonts() – Metoda do utworzenia czcionek.
- Private void createGraphics() – Metoda do tworzenia wszystkich elementów graficznych gry.
- Private void loadAssets() – Metoda do ładowania wszystkich zasobów gry.
- Private void startRotateSound() – Metoda do odtwarzania dźwięku obrotu wieżyczek.
- Private void playShootSound() – Metoda do odtwarzania dźwięków wystrzałów.
- Public void readyButtonCheck() – Metoda do sprawdzenia czy wszystkie statki są na dobrych pozycjach i wystartowanie bitwy.
- Private void update(float) – Metoda do aktualizacji logiki gry.
- Public void render(float) – Metoda do renderowania całej szaty graficznej gry.
- Public void show() – Metoda wywoływana po utworzeniu ekranu gry.
- Public void pause() – Metoda obsługująca pauzę aktywności okna w oknie ekranu gry.
- Public void resume() – Metoda obsługująca wznowienie aktywności okna w oknie ekranu gry.
- Public void resize(int , int) – Metoda obsługująca zmianę rozmiaru okna w oknie ekranu gry.
- Public void hide() - Metoda obsługująca ukrycie okna w oknie ekranu gry.
- Public void dispose() - Metoda obsługująca niszczenie elementów silnika libgdx oraz okna ekranu gry.
- Public boolean keyDown(int) – Metoda do obsługi gdy klawisz zostanie wciśnięty.

- Public boolean keyUp(int) - Metoda do obsługi gdy klawisz zostanie puszczony.
- Public boolean keyTyped - Metoda do obsługi gdy klawisz zostanie kliknięty
- Public boolean touchDown(int , int , int , int) – Metoda wywoływana gdy klawisz myszki zostanie wciśnięty.
- Public boolean touchUp(int , int , int , int) – Metoda wywoływana gdy klawisz myszki zostanie puszczony.
- Public boolean touchDragged(int ,int , int) – Metoda wywoływana gdy mysz została poruszona po ekranie gdy jej klawisz jest wciśnięty.
- Public boolean mouseMoved(int , int) – Metoda wywoływana gdy mysz została poruszona po ekranie.
- Public boolean scrolled(float , float) – Metoda wywoływana gdy scroll myszy zostanie użyty.

Praca wykonana przez członków zespołu:

- *Dominik Grudzień* – Twórca Animatora , okien menu głównego i dialogowych , sliderów , interfejsu , przycisków tesktowych i graficznych.
- *Patryk Grzywacz* – Twórca assetów , efektów dźwiękowo - graficznych, obiektów i mechanik rozgrywki gry , „sztucznej inteligencji" i testu Junit.
- *Adrian Pełka* – Twórca wyników, obróbki dźwięków i assetów , mechanizmów rozmieszczeń i kontroli działań.

Wszyscy członkowie zespołu pracowali , tworzyli , testowali i debugowali aplikację aż do wersji 1.2 oraz tworzyli dokumentację.