
**СУ “Св. Климент Охридски”,**
**ФМИ – Софтуерно инженерство**
**Курсов проект по Мобилни приложения**

# Starship

*Йоан Евгениев Стоянов, Факултетен № 61830*

# Съдържание

# 1. Въведение

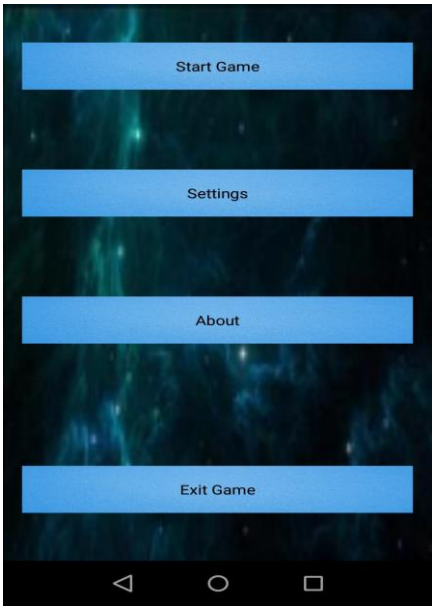
Приложението наподобява аркадната игра Galaga, но в опростен вариант. В тази игра трябва да се унищожат вражеските кораби като всяко ниво става все по-трудно и по-трудно.

Приложението разполага с едно главно Activity(*MenuActivity*), от което се достъпват другите Activity-та: *SpaceshipActivity*(започване на играта), *SettingsActivity*(настройки на играта), *AboutActivity*(информация за играта), чрез съответните бутони и бутон за изключване на приложението.

# 2. Описание на главните activities.

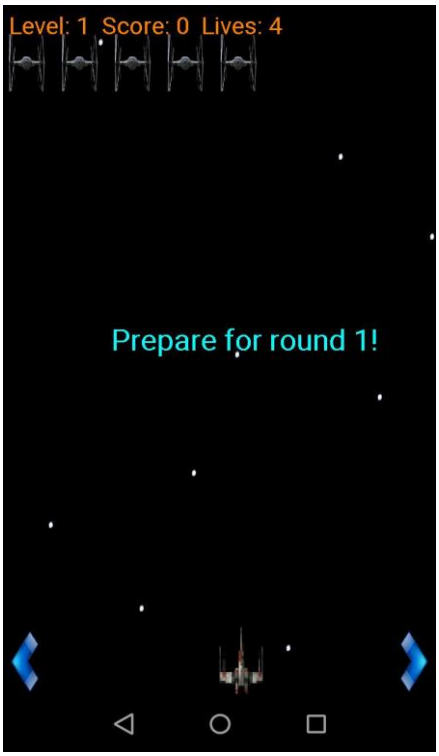
### 2.1.1. MenuActivity

- 4 x Button – бутони съответно за различните activity-та.




### 2.1.2. SpaceshipActivity

- SpaceshipView – собствен view наследен от SurfaceView, в който се намира основата на играта. Използва се SurfaceView, за да се изрисуват анимациите върху екрана.



### 2.1.3. SettingsActivity

- 3 x Button – бутони съответно за нулиране на резултата на играча, промяна на псевдонима на играча и връщане към главното меню
- 1 x Dynamic Button – бутон, който се появява, когато е натиснат бутона за смяна на псевдонима.
- 1 x Dynamic TextView – текстово поле, което се появава, когато е натиснат бутона за смяна на псевдонима.


### 2.1.4. AboutActivity

- 1 x TextView – текстово поле, което съдържа информация за играта.
- 1 x Button – бутон за връщане към главното меню.

# 3. Описание на класовете

## 3.1. Playership

### 3.1.1. Полета - RectF rect, Bitmap bitmap, float length, float height, float x, float y, float shipSpeed, int shipMoving;

### 3.1.2. Статични константи – STOPPED, LEFT, RIGHT;

### 3.1.3. Публични методи

3.1.3.1. PlayerShip(Context context, int screenX, int screenY) – конструктор,в който се инициализира кораба на играча(размери, картинка);
3.1.3.2. RectF getRect(), Bitmap getBitmap(), float getX(), float getLength(), float getHeight, int getMovementState() – getters;
3.1.3.3. void setMovementState(int state) – setter
3.1.3.4. void update(long fps) – променя местоположението на кораба в зависимост от това дали трябва да се движи наляво или надясно.

## 3.2. Enemy

### 3.2.1. Полета – RectF rect, Random generator = new Random(), Bitmap bitmap, float length, float height, float x, float y, float shipSpeed, int shipMoving, Boolean isVisible;

### 3.2.2. Константи – LEFT, RIGHT, SHIP_SPEED;

### 3.2.3. Публични методи

3.2.3.1. Enemy(Context context, int row, int column, int screenX, int screenY) – конструктор, в който се инициализират вражеските кораби( размери,картинка, на коя позиция във формацията от кораби да са);
3.2.3.2. boolean getVisibility(), RectF getRect(), Bitmap getBitmap(), float getX(), float getY(), float getLnegth() – getters;
3.2.3.3. void setInvisible() – setter;
3.2.3.4. void update(long fps) – променя местоположението на кораба в зависимост от това дали трябва да се движи наляво или надясно;
3.2.3.5. void reverse() – задава на изкуствения интелект (вражеските кораби), ако стигнат до единия край на екрана да тръгнат в обратната посока;
3.2.3.6. boolean takeAim(float playerShipX, float playerShipLength, int difficultyLevel) – алгоритъм за това кога изкуствения интелект да стреля по играча. Ако корабът на играча се намира срещу вражески кораб, тогава шанса за изстрел се увеличава. Също така след всяко ниво, шансът вражески кораб да стреля се увеличава;

## 3.3. Star

### 3.3.1. Полета – float x, float y, float width, float height, RectF rect, Bitmap bitmap, Random generator = new Random(), int starSpeed;

### 3.3.2. Константи – MINIMUM_STAR_SPEED, STAR_SPEED_THRESHOLD;

### 3.3.3. Публични методи

3.3.3.1. Star(Context context, int numStar, int screenX, intscreenY) – конструктор,в който се инициализират звездите като всяка получава произволно местоположение и скорост, за да може така да се симулира космос;
3.3.3.2. Bitmap getBitmap(), float getX(), float getY() – getters;
3.3.3.3. void update(long fps) – променя местоположението на звездата;
3.3.3.4. void resetPosition() – ако звездата стигне до края на екрана, то тя се връща в началото;

### 3.4. Blaster

#### 3.4.1. Полета – float x, float y, RectF rect, int heading, float speed, int width, int height, Boolean isActive;

#### 3.4.2. Константи – UP, DOWN;

#### 3.4.3. Публични методи

3.4.3.1. Blaster(int screenY) – конструктор, в който се инициализира изстрел като се задава да е неактивен.
3.4.3.2. RectF getRect(), Boolean getStatus() – getters;
3.4.3.3. float getImpactPointY() – връща допирната точка на изстрела,в зависимост от това дали е бил изстрелян от играча или от вражески кораб;
3.4.3.4. void setInactive() – прави неактивен изстрела;
3.4.3.5. boolean shoot(float startX, float startY, int direction) – задейства се изстрел като се задават координатите му и в коя посока да е изстрелян, но ако вече е активен(изстрелян),тогава не се прави нищо;
3.4.3.6. void update(long fps) – променя се местоположението на изстрела, в зависимост дали е изстрелян нагоре или надолу;

### 3.5. Controls

#### 3.5.1. Полета – Bitmap bitmap, float length, float height, float x, float y;

#### 3.5.2. Публични методи

3.5.2.1. Controls(Context context, int screen, int screen, Boolean isLeft) – конструктор, в който се инициализират бутоните за движение на играча.
3.5.2.2. float getLength(), float getX(), float getY(), Bitmap getBitmap() – getters;

### 3.6. SpaceshipView extends SurfaceView implements Runnable

#### 3.6.1. Полета – Context context, Thread gameThread = null, SurfaceHolder ourHolder, volatile boolean playing, boolean paused = true, boolean startedRound = false, Canvas canvas, Paint paint, Controls leftArrow, Controls rightArrow, long fps, long timeThisFrame, int screenX, int screenY, Star[] stars = new Star[25], Playership playership, Blaster[] blasters = new Blaster[10], Blaster[] enemyBlasters = new Blaster[200], int nextBlaster = 0, int nextBlasterEnemy = 0, Enemy[] enemies = new Enemy[NUMBER_OF_MAXIMUM_ENEMIES], int enemyCounter = 0, int numEnemies = 0, int score = 0, int maxScore = 0, int level = 0, int lives = 4,long timeShotFired =0, int columnMax, int rowMax, boolean printRoundMessage = false, boolean printLostMessage = false, boolean printWonMessage = false, boolean gameLost = false, boolean lost = false, boolean won = false, boolean bumped = false, long startingTime = 0;

#### 3.6.2. Константи – LEFT, RIGHT, TIME_BETWEEN_SHOTS, NUMBER_OF_MAXIMUM_ENEMIES

#### 3.6.3. Private методи

3.6.3.1. void prepareStars() – инициализира масив от звезди;
3.6.3.2. void preparePlayerShip() – инициализира корабът на играча;
3.6.3.3. void preparePlayerBlasters() – инициализира изстрелите на играча;
3.6.3.4. void prepareEnemeyBlasters() – инициализира вражеските изстрели;
3.6.3.5. void prepareEnemies() – инициализира вражеските кораби;
3.6.3.6. void prepareLevel() – пригатвя играта за следващото ниво като нулира позициите на корабите. Има и проверка ако играчът мине всички нива, играта да приключи;

3.6.3.7. void updateStars() – променя местоположението на звездите като прави проверка дали не стигат края на екрана;
3.6.3.8. void updatePlayerShip() – променя местоположението на кораба на играча като прави проверка дали корабът не излиза от екрана;
3.6.3.9. void updateEnemies() – променя състоянието на вражеските кораби: тяхното местоположение, дали ще стрелят, проверка дали не излизат от екрана;
3.6.3.10. void updatePlayerBlasters() – променя състоянието на изстрелите на играча: тяхното местоположение;
3.6.3.11. void updateEnemyBlasters() – променя състоянието на вражеските изстрели: тяхното местоположение;
3.6.3.12. void updatePlayerEnemyCollision() – прави проверка дали изстрел от кораба на играча се е сблъскал с вражески кораб: ако е така, вражеския кораб е унищожен и се увеличава резултата на играча. При унищожаване на текущата вълна от врагове се започва подготовка за следващото ниво.  Проверката се прави чрез метода на RectF intersects;
3.6.3.13. void updateEnemyPlayerCollision() – прави проверка дали изстрел от вражески кораб се е сблъскал с кораба на играча: ако е така, се намаляват животите на играча. Когато животите на играча станат 0, играта приключва. Проверката се прави чрез метода на RectF intersects;
3.6.3.14. void update() – тука се правят всички промени на състоянита на обектите;
3.6.3.15. void returnToMenu() – задейства се процедурата по връщане към главното меню след като е приключила играта;
3.6.3.16 void draw() – този метод рисува обектите върху екрана чрез canvas I paint. Първо прави проверка дали е валидна рисувачната повърхност. За съобщенията за известяване(ако е свършило нивото или играчът е спечелил/загубил) има изчакване от 5 секунди, в които играта е паузирана;

### 3.6.4. Публични методи

3.6.4.1. SpaceshipView(Context context, int x, int y) – конструктор, в който се инициализира самата игра;
3.6.4.2. void run() – в този метод се прави проверка ако играта е в режим на игране да се актуализират всички обекти и да се нарисуват върху екрана. Също така се изчисляват и с колко кадъра върви играта;
3.6.4.3 boolean onTouchEvent(MotionEvent motionEvent) – проверява се входа от потребителя(коя част на екрана е натиснал) и се извършват съотвените действия: стрелките за преместване на кораба, всяка останала част на екрана за произвеждане на изстрел;

## 4. Използвани технологии

4.1. Genymotion 2.6.0
4.2. Android Studio 2.1