

# Dokumentácia zápočtového programu

## Názov:

Vesmírna strelačka


## Autor:

Juraj Serafin

## Popis:

2D strelačka vesmírnych lodí. Po spustení sa zobrazí okno s možnosťou výberu herného módu. Hráčova loď sa pohybuje v spodnej časti obrazovky po X –ovej osi a strela projektily na flotily nepriateľských lodí nad ňou. Úlohou hráča je jednotlivé lode ničiť prostredníctvom vystrelených projektilov a tým zvyšovať svoje skóre, a zbierať hernú menu. Po nadobudnutí určitých mét skóre má hráč možnosť svoju loď vylepšovať v rôznych aspektoch za získanú menu. Hra beží donekonečna, skončí sa po vyčerpaní počtu životov po zásahoch nepriateľských lodí, prípadne ukončením aplikácie po stlačení tlačidla **Exit**, ktoré sa zobrazí po stlačení tlačidla s nastaveniami.

## Návod na použitie:

1. Spustíte aplikáciu.
2. Zvoľte herný mód.
  - 3.1. V prípade že ste zvolili mód **Manual**, pre pohyb lode použite tlačidlá **A** a **D**, alebo tlačidlá **dol'ava** a **doprava**, pre strelbu použite **spacebar** a pre jednotlivé typy vylepšení tlačidlá **1,2,3**.
  - 3.2. V prípade, že ste zvolili mód **Auto**, sa loď bude ovládať automaticky.
4. V prípade, že chcete hru pozastaviť, alebo ju ukončiť stlačte tlačidlo  a vyberte príslušnú možnosť.

## **Celkový postup lode pri automatickom móde:**

1. Loď zameria najbližšieho nepriateľa.
2. Loď skontroluje či sa na obrazovke nachádza minca.
3. V prípade, že je minca na obrazovke nastaví jej polohu ako cieľ pohybu, v prípade, že nie, loď sa bude pohybovať k najbližšiemu nepriateľovi.
4. Pred každým pohybom sa loď presvedčí, či je daný pohyb bezpečný, v prípade, že áno sa pohyb uskutoční, v opačnom prípade loď vykoná pohyb, ktorý je výsledkom funkcie *FindSafe Move()*
5. Loď v priebehu pohybu páli projektily v prípade, že je schopná paľby a bude pripravená páliť po nadobudutí cieľa pohybu.
6. V prípade získania mince loď automaticky vylepší počet životov ak je  $\leq 3$ .

V opačnom prípade vylepšuje rovnomerne rýchlosť pohybu a rýchlosť projektilu.

## **Technická časť**

### Obsah - Hlavné časti

Technická časť .....	2
Bars and Labels .....	4
HealthBar.cs .....	5
ScoreLabel.cs .....	5
UpgradeMessage.cs .....	5
Entities .....	5
AbstractEntity.cs .....	5
IsOnScreen(PictureBox pictureBox) .....	5
GetBackUp() .....	5
CausedByEnemy(string hitObjectTag, PictureBox pictureBox) .....	6
GotHit(string hitObjectTag) .....	6
DeleteObject() .....	6
LocateHero() .....	6
AimForHero() .....	6

GetRandomTarget().....	6
GetMoveDirection().....	6
GetRandomMoveDestination() .....	6
PrepareMovingToHero() .....	6
GetDistance() .....	6
GetShifts(int distance, int speed, Vector2d directionVector) .....	7
InitializeProjectile(Projectile projectileToSet) .....	7
IsPositionToShoot().....	7
Shoot(object? sender, EventArgs e) .....	7
MoveStraight(PictureBox pictureBox, Vector2d shifts) .....	7
MoveCurvy(PictureBox pictureBox, Vector2d shifts) .....	7
Initialize().....	7
InitializeShootingTimer().....	7
IsAlive() .....	7
InitializePictureBox().....	7
BasicEnemyEntity.cs.....	8
MidEnemyEntity.cs .....	8
HardEnemyEntity.cs.....	8
BossEnemyEntity.cs .....	8
HeroEntity.cs .....	8
MoveManual().....	8
Move().....	8
FireReady().....	8
Death().....	9
PerformUpgrade(int upgradeCode).....	9
Action() .....	9
CanShoot() .....	9
CalculateNextMove().....	9
FindClosestEnemy().....	9
SetClosestEnemyData(PictureBox pictureBox).....	10
CompareEnemies(string name1, string name2) .....	10
CalculateMoveShootDistance(PictureBox pictureBox).....	10
DislocateDead().....	10
AreaSafe(int moveToMake).....	10

DangerInWay(PictureBox Area) .....	10
CoinAvailable() .....	10
FindSafeMove() .....	10
UpgradeEvenly() .....	11
Game objects .....	11
Coin.cs .....	11
Projectile.cs .....	11
TravelUp() .....	11
MoveStraight() .....	11
Spawning .....	11
EnemyController.cs .....	11
CreateEnemy() .....	12
SpawnInitialEnemyGroup(int amount, int enemyDifficulty) .....	12
SpawnInitialEnemyWave() .....	12
SpawnWave() .....	12
IncreaseDifficulty() .....	12
NextEnemyPicker.cs .....	13
GameWindow.cs .....	13
MainEvent(object sender, EventArgs e) .....	13
KeyIsDown(object? sender, KeyEventArgs e), KeyIsUp(...) .....	13
SettingsButton_Click(object sender, EventArgs e) .....	13
MainMenu.cs .....	14
SettingsForm.cs .....	14
images .....	14
Možné rozšírenia: .....	14
Alternatívne riešenia .....	14
Zhodnotenie .....	15

## Bars and Labels

**Funkcia:** Kontrola a ovládanie herných počítačov a správy o vylepšení lode.

## HealthBar.cs

Trieda definujúca počítateľ života, obsahuje len metódy *Initialize()* na definovanie vlastností premennej *healthLabel* a *Update()* na aktualizáciu ukazovateľa počtu pri zásahu, alebo vylepšení života.

## ScoreLabel.cs

Trieda definujúca počítateľa skóre. Okrem metód *Initialize()* a *Update()* (s analogickým významom ako pri triede *HealthBar* obsahuje aj metódu *CheckUpgrade()* na kontrolu dostupnosti vylepšenia a následné zvýšenie obtiažnosti.

## UpgradeMessage.cs

Určená na zobrazenie správy o dostupnosti vylepšenia.

## Entities

**Funkcia:** Ovládanie a výpočtové operácie pre lode

## AbstractEntity.cs

Abstraktná trieda na definujúca všeobecné nastavenia a metódy pre všetky lode. Ako argument konštruktoru berie premennú typu **Form**, na ktorom bude jej **PictureBox** existovať.

## IsOnScreen(PictureBox pictureBox)

Na základe súradníc skontroluje či sa premenná, ktorú dostane v argumente nájde na obrazovke.

## GetBackUp()

Využíva funkciu *IsOnScreen* na detekciu prítomnosti premennej *icon* na obrazovke, a v prípade výsledku **false** ju vráti späť na vrchnú časť.

**CausedByEnemy**(string hitObjectTag, PictureBox pictureBox)

Kontrola, či *pictureBox*, bol zasiahnutý iným **pictureBoxom**, ktorého vlastnosť **tag** sa zhoduje s argumentom *hitObjectTag*.

**GotHit**(string hitObjectTag)

Prejde všetky **pictureBoxy** z vlastnosti premennej *screen Controls*, skontroluje, či má prienik a premennou *icon* a v prípade že áno, vymaže *icon* zo *screen.Controls*, nastaví hodnotu vlastnosti *Dead* na **true** a vráti hodnotu **true**.

**DeleteObject**()

Metóda na odstraňovanie eliminovaných lodí z formulára. Ukončí *Timer* na strieľanie a pohyb, vymaže *icon* zo *screen.Controls*, nastaví hodnotu vlastnosti *Dead* na **true**.

**LocateHero**()

Načíta polohu hrdinu do zložiek vektoru *heroLocation*.

**AimForHero**()

Zavolá funkciu *LocateHero* a určí hrdinu ako cieľ streľby – priradí hodnoty polohy **X** a **Y** do vlastností príslušných vlastností vektoru *shootTargetCoordinates*.

**GetRandomTarget**()

Určí náhodný cieľ streľby.

**GetMoveDirection**()

Vypočíta vektor určený na pohyb lode.

**GetRandomMoveDestination**()

Určí náhodný cieľ pohybu.

**PrepareMovingToHero**()

Pomocou príslušných funkcií. Načíta potrebné súradnice na pohyb k hrdinovi.

**GetDistance**()

Vypočíta vzdialenosť k cieľu pohybu.

**GetShifts**(int distance, int speed, Vector2d directionVector)

Do premennej na základe hodnôt z argumentu vypočíta hodnoty vektoru, ktoré budú určovať vzdialenosti na osi **X** a **Y**, ktoré bude **pictureBox** pri pohybe chcem prekonávať.

**InitializeProjectile**(Projectile projectileToSet)

Inicializuje projektil a nastaví mu potrebné hodnoty na pohyb a identifikáciu.

**IsPositionToShoot**()

Cieľom je skontrolovať, či sa loď nenachádza príliš blízko hrdinu, aby bol hrdina schopný vyhnúť sa vypálenému projektilu.

**Shoot**(object? sender, EventArgs e)

Inicializuje projektil, ktorý tak bude mať nastavené potrebné parameter a začne sa pohybovať.

**MoveStraight**(PictureBox pictureBox, Vector2d shifts)

Uskutoční pohyb po rovnej čiare.

**MoveCurvy**(PictureBox pictureBox, Vector2d shifts)

Uskutoční kľukatý pohyb.

**Initialize**()

Inicializuje premennú *icon* a nastaví vektory pre pohyb.

**InitializeShootingTimer**()

Nastaví a spustí **Timer** pre strieľanie.

Analogicky **InitializeLifeTimer**()(kontrola života) a **InitializeMovingTimer**()

**IsAlive**()

Na základe vlastnosti **Tag** nepriateľského projektilu zistí, či bola loď zasiahnutá a v prípade, že jej vlastnosť *Health* = 0, aktualizuje skóre, zavolá metódu *DeleteObject* a vymaže **pictureBox** z obrazovky.

**InitializePictureBox**()

Nastaví potrebné vlastnosti premennej *icon*.

### **BasicEnemyEntity.cs**

Trieda definujúca najslabší typ nepriateľskej lode. Loď sa pohybuje na náhodné miesto v spodnej časti obrazovky na rovnej čiare bez strelby (podľa toho odvodí funkcie). Dedí takmer všetky potrebné funkcie z *AbstractEntity*, s výnimkou *StartOperating()*. (absencia strelby)

### **MidEnemyEntity.cs**

Trieda definujúca nepriateľskú loď stredne ťažkej obtiažnosti. Pohybuje sa rovno na náhodné miesto v dolnej časti obrazovky, na rozdiel od slabšej lode však aj strieľa.

### **HardEnemyEntity.cs**

Trieda definujúca nepriateľskú loď ťažkej obtiažnosti. Loď zameria svoj pohyb a strelbu na polohu hrdinu a pohybuje sa za ním po rovnej čiare.

### **BossEnemyEntity.cs**

Trieda definujúca najsilnejšiu nepriateľskú loď. Loď každý pohyb a výstrel zamieri na polohu hrdinu. Pohybuje sa klukatým pohybom a má väčšiu veľkosť projektílov.

### **HeroEntity.cs**

Trieda definujúca vlastnosti hráčovej lode a funkcie umožňujúce automatický a manuálny herný mód.

#### **MoveManual()**

Funkcia zaručujúca manuálny pohyb a schopnosť lode ostať na obrazovke po nadobudnutí krajných bodov, využíva na to funkcie *MoveRight()*, *MoveLeft()*.

#### **Move()**

Pomocou funkcie *IsAlive()* určí či má loď pozitívny počet životou a na základe hodnoty vlastnosti *AutoModeOn* volá príslušné funkcie na manuálne a automatické pohyby lode.

#### **FireReady()**

Skontroluje či ubehla perióda času, ktorá by umožňovala výstrel.



### **Death()**

Uberie život a skontroluje počet životov, ak je rovný 0, nastaví premennú *Dead* na true a hra končí.

### **PerformUpgrade(int upgradeCode)**

Na základe hodnoty z argumentu zavolá príslušnú funkciu na vylepšenie lode.

### **Action()**

Určuje akciu lode pri automatickom móde. Na základe dostupnosti mince na obrazovke a možnosti streľby rozhodne o pohybe a streľbe.

### **CanShoot()**

Určí či má loď možnosť vystreliť tak, aby bola pripravená na strľbu pri dosiahnutí cieľovej lokácie.

### **CalculateNextMove()**

Zavolá funkciu *FindClosestEnemy()*, a na základe vektora určujúceho pohyb rozhodne o ďalšom pohybe.

### **FindClosestEnemy()**

Nájde najbližšiu nepriateľskú loď v zmysle počtu pohybov lode a pohybov projektilu.

**SetClosestEnemyData(PictureBox pictureBox)**

Nastaví príslušné hodnoty určujúce údaje o najbližšom nepriateľovi, prípadne ich aktualizuje ak sa našiel bližší, alebo rovnako blízky, ale silnejší nepriateľ, ktorého je efektívnejšie eliminovať prvého. V prípade zostrelenia najbližšieho nepriateľa, z neho odstráni zameranie.

**CompareEnemies(string name1, string name2)**

Porovná lode na základe vlastností ich mena.

**CalculateMoveShootDistance(PictureBox pictureBox)**

Vypočíta vzdialenosť v zmysle počtu pohybov lode a pohybov projektilu premennej z argumentu.

**DislocateDead()**

Odstráni zameranie zo zameraného najbližšieho nepriateľa.

**AreaSafe(int moveToMake)**

Skontroluje či je pohyb o počet pixelov z argumentu bezpečný – loď po pohybe nebude mať prienik s nepriateľským objektom.

**DangerInWay(PictureBox Area)**

Skontroluje, či je premenná z argumentu v prieniku s ktorýmkoľvek nepriateľským objektom.

**CoinAvailable()**

Skontroluje, či je na obrazovke dostupná minca a ak áno, zaznamená jej polohu na osi X.

**FindSafeMove()**

Nájde bezpečný pohyb. (ak taký existuje)

## UpgradeEvenly()

Automaticky vylepší loď rovnomerne v oblasti rýchlosti pohybu a projektilu.

## Game objects

**Funkcia:** Definícia tried herných predmetov projektil a minca. Hlavná úloha metód je inicializácia, umiestnenie, pohyb, odstránenie z obrazovky a identifikácia prieniku s **PictureBoxom** hrdinu

### Coin.cs

Definuje triedu hernej meny, po ktorej získaní, hráč získa možnosť umožní vylepšiť jeden z troch aspektov lode: rýchlosť (*Speed*), rýchlosť pohybu projektilu (*ProjectileSpeed*), počet životov (*Health*). Argument jej konštruktoru je inštancia triedy odvodenej od triedy **Form**. Na inicializáciu, umiestnenie, odstránenie z obrazovky využíva podobné metódy ako už uvedené triedy, funkcia **PickedUp()** slúži na identifikáciu prítomnosti hrdinu v oblasti mince.

### Projectile.cs

#### TravelUp()

Metóda definujúca pohyb náboja hrdinu po priamke rovnobežnej s osou **Y**.

#### MoveStraight()

Metóda, ktorá slúži na pohyb projektilu po priamke definovanej vektorom *directionVector*.

## Spawning

### EnemyController.cs

**Funkcia:** Spawning, kontrola **Timerov** nepriateľských lodí. Argument jej konštruktoru je inštancia triedy odvodenej od triedy **Form**.

## **CreateEnemy()**

Podľa hodnoty z argumentu určí typ nepriateľa, ktorý bude vytvorený.

## **SpawnInitialEnemyGroup(int amount, int enemyDifficulty)**

Vytvorenie skupiny nepriateľov rovnakej obtiažnosti pri prvej vlne. Novo vytvorené inštancie triedy odvodenej od *AbstractEntity* umiestní do zoznamu *\_enemies*.

## **SpawnInitialEnemyWave()**

Vytvorenie prvej vlny nepriateľov, počty jednotlivých typov na jednotlivých riadkoch obrazovky sú špecifikované v zozname *NumsOfEnemiesOnLines*.

## **SpawnWave()**

Vytvorenie neskorších vln nepriateľov. Typy nepriateľov sú určené na základe šance na spawn, ktorú majú jednotlivé typy špecifikované v zozname *EnemyDifficultyChanceToSpawn*, súradnice, na ktorých sa objavajú sú však určené náhodne.

## **MoveEnemy()**

Odstráni eliminované entity zo zoznamu *\_enemies*, nájde najmenší index v zozname *\_enemies*, ktorý patrí aktívnej lodi a inicializuje jeho **Timery**.

## **IncreaseDifficulty()**

Zvýši šance na spawn pre silnejšie lode. Uberie šancu na spawn slabšej lodi a rozdelí ju medzi silnejšie. Nakoniec zvýši rýchlosť spawningu a uvádzania lodí do pohybu prostredníctvom metódy *QuickenSpawning()*.

## NextEnemyPicker.cs

**Funkcia:** Výber typu nepriateľskej lode, ktorá sa spawnne na základe šance na spawn

definovanej v zozname *EnemyDifficultyChanceToSpawn*. Funkcie *CalculateProbability(List<(int, int)> valueProbabilityDoubles)* a *ValidProbability(List<(int, int)> valueProbabilityDoubles)* slúžia na overenie správneho formátu hodnôt pravdepodobnosti spawnu jednotlivých položiek v zozname *EnemyDifficultyChanceToSpawn*.

## GameWindow.cs

Hlavný formulár aplikácie. Trieda odvodená od triedy **Form**. Obsahuje funkcie na pohyb a strelbu hrdinu pri manuálnom móde hry, ovládanie premenných typu *HealthBar* a *ScoreLabel* na počítanie počtu životov a skóre,


**MainEvent**(object sender, EventArgs e)

Umožňuje pohyb hrdinu a kontroluje hodnotu skóre, následné objavenie mince pri dosiahnutí dotyčnej méty a vylepšenie lode.

**KeyDown**(object? sender, KeyEventArgs e), **KeyUp**(...)

Nastavujú potrebné vlastnosti hrdinu na to, aby bol hrdina schopný pohybu doprava, doľava a zaistujú konkrétne vylepšenie na základe stlačenia príslušného tlačidla na klávesnici.


**SettingsButton\_Click**(object sender, EventArgs e)

Po stlačení  spustí inšanciu triedy *SettingsForm* odvodenej od triedy **Form**, ktorá ponúkne užívateľovi ukončiť hru, prípadne sa do nej vrátiť. – metóda *Resume()*.

## MainMenu.cs

Trieda odvodená od triedy **Form** reprezentujúci štartové menu, ktoré umožňuje užívateľovi zvoliť herný mód prostredníctvom metód *ManualModeButton\_Click(object sender, EventArgs e)* a *AutoModeButton\_Click(object sender, EventArgs e)*.

## SettingsForm.cs

Trieda reprezentujúca formulár na ukončenie aplikácie alebo spätné vrátenie sa do hry po stlačení , čo zabezpečujú jej metódy *ExitButton\_Click(object sender, EventArgs e)* a *ResumeButton\_Click(object sender, EventArgs e)*.

## images

Obsahuje obrázky, použité na výplň pictureBoxov, tlačidiel a pozadí formulárov.

## Možné rozšírenia:

- Zdokonalenie umelej inteligencie, v smere rozpoznávania nepriateľov, čo by viedlo k možnosti predpokladať nasledujúci pohyb nepriateľa a pozíciu, na ktorú nepriateľ vystrelí
- Zlepšenie objektového návrhu pri herných objektoch
- Audiovizuálne efekty

## Alternatívne riešenia

- Pri programovaní jednotlivých lodí pridávať každej z nich jedinečný identifikátor, ktorý by bolo možné čítať z *Controls*, čo by umožnilo hrdinovi rozpoznávanie lode a predikciu pohybov, strelby
- Použiť polymorfizmus aj pri ostatných objektoch objavujúcich sa na obrazovke okrem lodí

## **Zhodnotenie**

Cieľ mojej práce je napriek drobným nedokonalostiam naplnený. Vďaka objektovému návrhu neobsahovalo programovanie nepriateľských lodí žiadne väčšie prekážky, čoho výsledkom bolo overriding len malého množstva funkcií. Rovnako aj spawning, kontrola nepriateľských lodí a herné predmety ako projektily a mince fungujú bezproblémovo. Obtiažnosť hry sa zvyšuje s pribúdajúcim skóre, kedy sa zvyšuje šanca na spawn silnejších nepriateľov. Hráčovi ale vzápätí pribúdajú vylepšenia, čoho dôsledkom je spravodlivý herný systém.