### 14.05.2018 Wrocław

# PROGRAMOWANIE OBIEKTOWE

INEW0003P

Projekt

Wydział Elektroniki	Kierunek: Informatyka
Grupa zajęciowa: Środa 17:05-18:45 E00-93bc	Semestr: 2017/18 LATO
Nazwisko i Imię: Stepanenko Nikita	Nr indeksu: 245816
Nazwisko i Imię: Matviichuk Dmytro	Nr indeksu: 240545
Nr. grupy projektowej: 7	
Prowadzący: mgr inż. Adam Włodarczyk	1

TEMAT:

Gra w czołgi

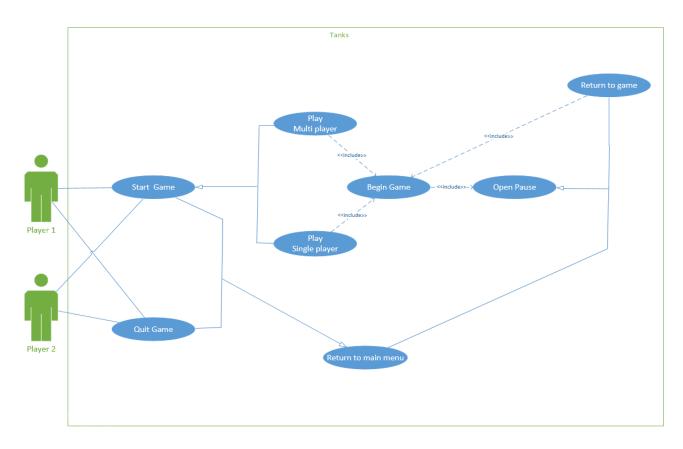
OCENA:	PUNKTY:
Data:	

# Założenia i opis funkcjonalny programu

To jest prosta interpretcja gry "Battle City" dla PC, w którą można glać pojedynczo oraz w dwóch. Sterowanie w menu zachdzi za pomocą muszy, a sterowanie czołgiem za pomocą <WASD+C> i <UP DOWN LEFT RIGHT + / >(w singleplayer zachodzi i to i to, a w multiplayer - zalieży od gracza) i ESC dla pause.

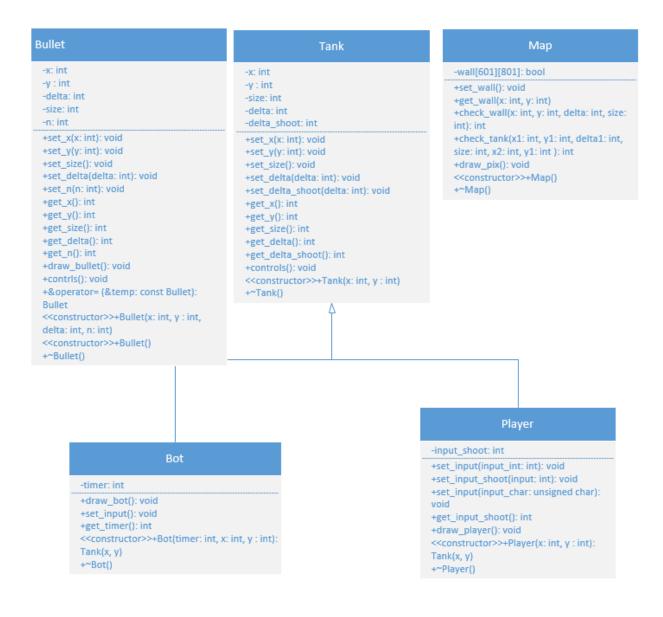
### Diagramy UML

Przypadków użycia:



#### Klas:

### C++



#### Java

#### Bullet Мар Tank -x: int -wall[601][801]: bool -x: int -y:int -y:int +set\_wall(): void -delta: int -size: int +get\_wall(x: int, y: int) -size: int -delta: int +check\_wall(x: int, y: int, delta: int, size: -n: int -delta\_shoot: int +set\_x(x: int): void +check\_tank(x1: int, y1: int, delta1: int, +set\_x(x: int): void +set\_y(y: int): void size: int, x2: int, y1: int ): int +set\_y(y: int): void +set\_size(): void +draw pix(): void +set\_size(): void +set\_delta(delta: int): void +set\_delta(delta: int): void <<constructor>>+Map() +set\_n(n: int): void +set\_delta\_shoot(delta: int): void +get\_x(): int +get\_x(): int Main +get\_y(): int +get\_y(): int +get\_size(): int +get\_size(): int +get\_delta(): int +get\_delta(): int +get\_n(): int +get\_delta\_shoot(): int +main(args: String[]): void +draw\_bullet(): void +load\_images(): void +controls(): void +contrls(): void <<constructor>>+Tank(x: int, y: int) <<constructor>>+Main() <<constructor>>+Bullet(x: int, y: int, delta: int, n: int) <<constructor>>+Bullet() Player Bot -input shoot: int -timer: int +set input(input: int): void +draw\_bot(): void +set\_input\_shoot(input: int): void +set\_input(): void +get\_input\_shoot(): int +get\_timer(): int +draw\_player(): void <<constructor>>+Bot(timer: int, x: int, y : int): <<constructor>>+Player(x: int, y : int): Tank(x, y) Tank(x, y) Single MyDisplay ~player1: ArrayList<Player> -texture\_menu: Texture -bots: int ~player2: ArrayList<Player> -texture\_pause: Texture -chanhe\_display: int -shoot\_timer: int -texture\_you\_win: Texture -old\_display: int ~bullet\_pl1: ArrayList<Bullet> ~player: ArrayList<Player> ~bullet\_pl2: ArrayList<Bullet> -texture\_game\_over: Texture ~bot: ArrayList<Bot> -texture\_player1\_win: Texture ~bullet\_pl: ArrayList<Bullet> +check\_bullet\_multi(x1: int, y1: int, +get\_change\_display(): int -texture\_player2\_win: Texture ~bullet\_bot: ArrayList<ArrayList<Bullet» +set\_change\_display(n: int): void delta1: int, size: int, n: int): int - texture\_bind\_map: Texture +check\_player\_multi(x1: int, y1: int, +checkFunc2(x1: int, y1: int, delta1: int, +get\_bots(): int +texture bind tank: Texture delta1: int, size: int): int size: int): int +check\_bullet\_single(x1: int, y1: int, +texture\_bind\_tank\_bot: Texture +checkFunc3(x1: int, y1: int, delta1: int, -shoot1(): void delta1: int, size: int, n: int): int +texture\_bind\_bullet: Texture -shoot2(): void size: int, n: int): int +check\_tank\_single(x1: int, y1: int, +load(): void +mouse\_button(): void -bullet\_draw(): void delta1: int, size: int): int +draw\_menu(): void +keyboard(x: int, y: int): void +show multi player(): void +shoot(): void +draw\_pause(): void +display(): void +keyboard\_player\_multi(x: int, y: int) +draw\_bullet\_player(): void +stats\_on\_0\_multi(): void +draw\_you\_win(): void +ticks(): void +draw\_bullet\_bot(nr: int): void +Clear\_Objects\_multi(): void +draw\_game\_over(): void +show\_single\_player(): void +draw\_player1\_win(): void +keyboard\_player\_single(): void

+stats\_on\_0\_single(): void

+Clear\_Objects\_single(): void

+draw\_player2\_win(): void

+bind\_map(): void

+bind\_tank(): void +bind\_tank\_bot(): void +bind\_bullet(): void

# Opis programu

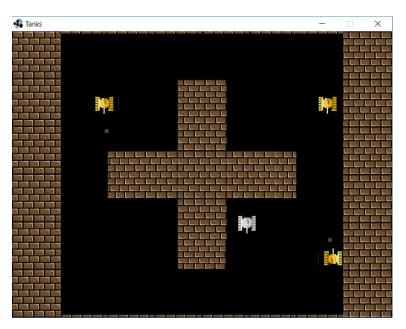
### Menu

W menu można wybrać grę pojedynczą lub dwuosobową oraz wyjść z gry. Sterowanie za pomocą myszy.



# Single

W single player gramy pojedynczo za pomocą <br/> <br/> WASD+C> i <br/> <br/> UP DOWN LEFT RIGHT + / >i ESC dla pause.



### Pause

W Pause można wrócić do gry lub do menu. Sterowanie za pomocą myszy.



### Multiplayer

sterowanie czołgiem za pomocą <WASD+C> i <UP DOWN LEFT RIGHT + /> (dla 1 i 2 gracza) i ESC dla pause.



### Game Over, You Win, Player 1 WIN, Player 2 WIN

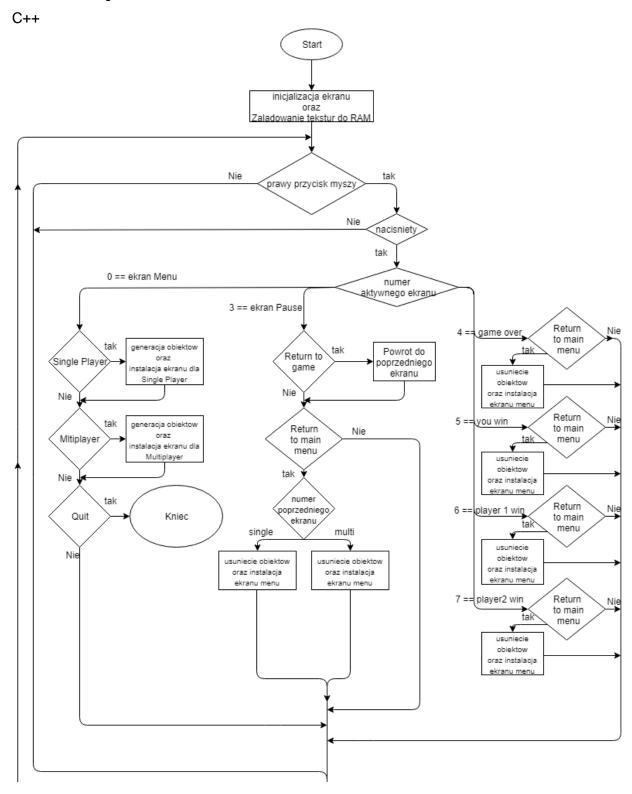
Nadpis zależy od wybranego typu gry i wyniku tej gry. RETURN TO MAIN MENU wraca do menu. Sterowanie y pomocą myszy

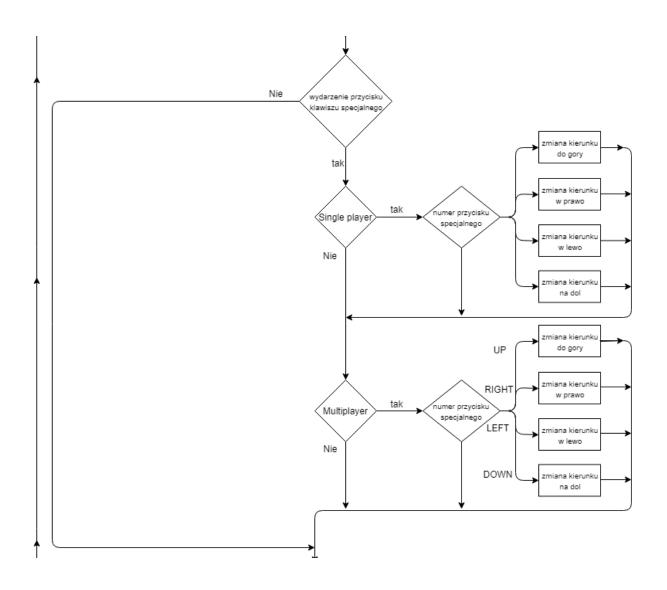


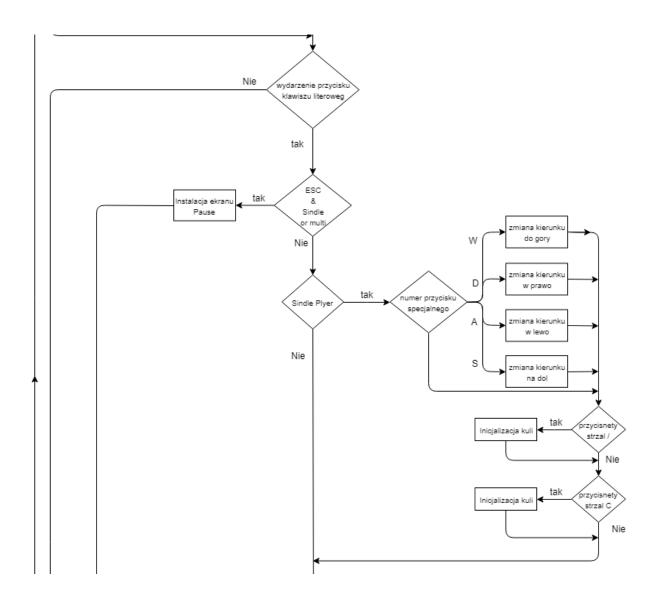
### Quit

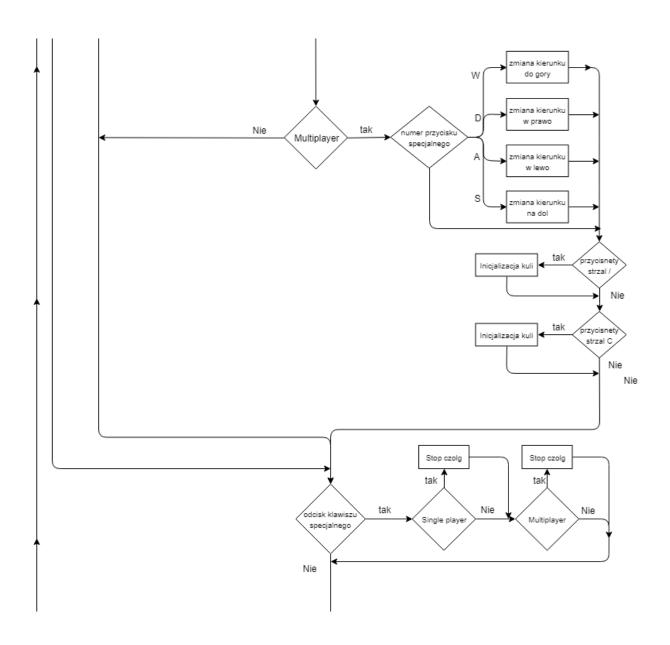
Po prostu wyjście z programu.

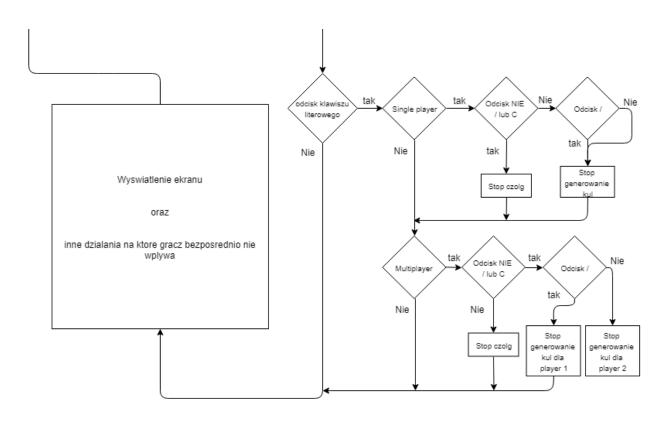
# Schematy blokowe

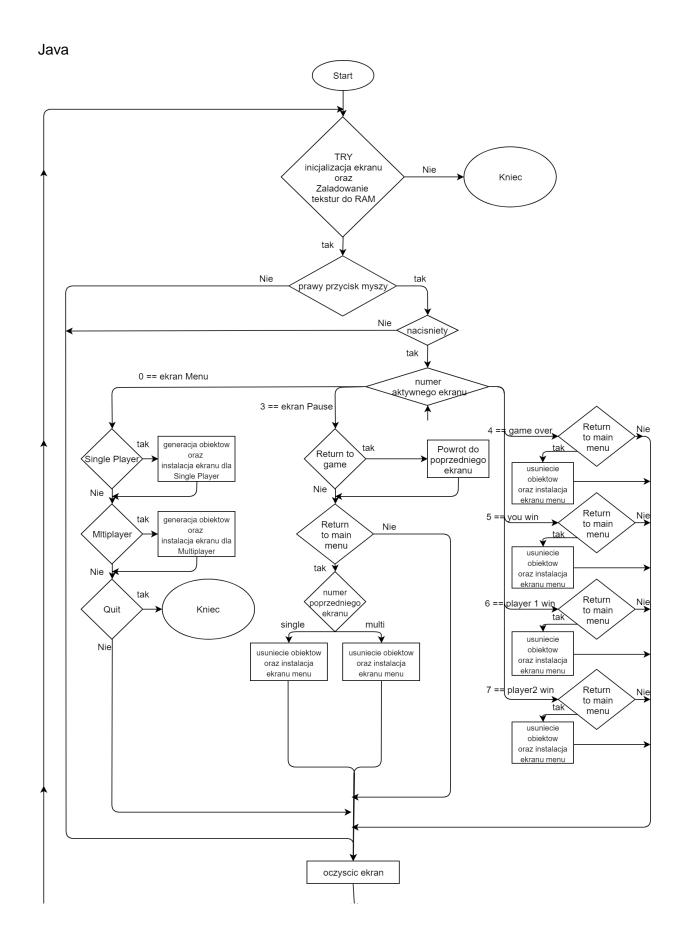


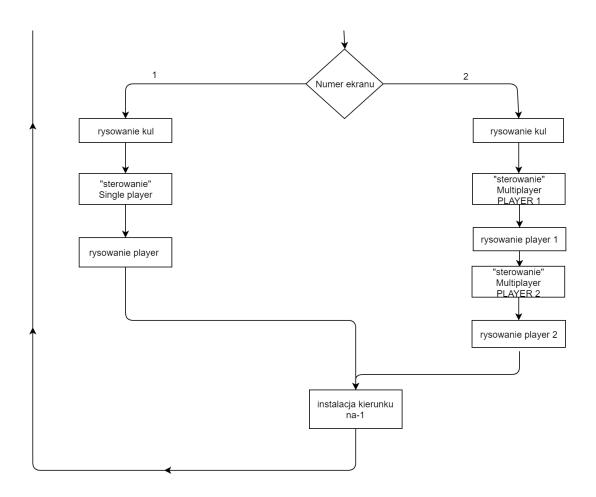




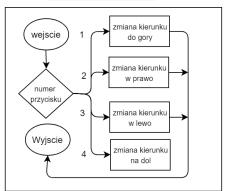




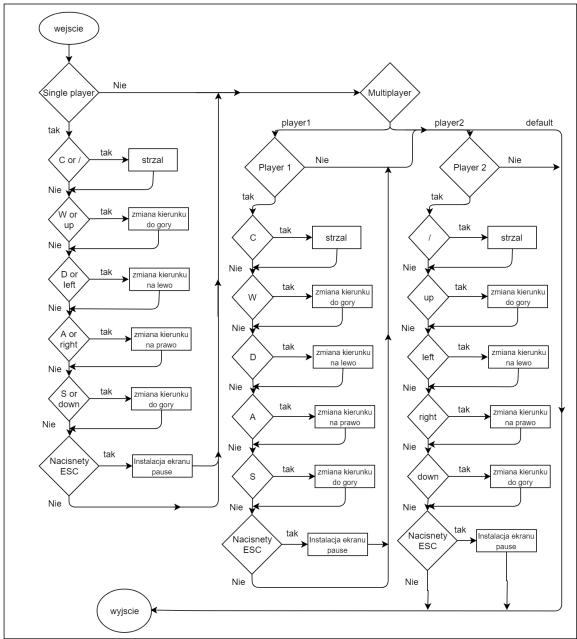




#### schemat zmiany kerunku...



Schemat "Sterowanie"



# Wnioski

Zaznajomiliśmy z działaniem Opengl (2D i 3D) w C++ oraz Java w różnych bibliotekach. Wybraliśmy opengl dlatego że z jego kożystano do dziś jako API oraz za pomocą go napisny Vulkan API.

Nie zdążyliśmy zrobić High score, bo dużo czasu straciliśmy na problemy związane z teksturami.