POLITECHNIKA ŚWIĘTOKRZYSKA

Wydział Elektroniki, Automatyki i Informatyki

PROGRAMOWANIE W JĘZYKU C2 – PROJEKT **SEMESTR ZIMOWY, 2018/2019 R.**

KIERUNEK:
INFORMATYKA

REALIZACJA:
ADRIAN JAKUBCZYK GR. 2ID12A
DOMINIK KALISZEWSKI GR. 2ID12A

TEMAT PROJEKTU:
Gra 2D w stylu „Worms”

1. Tematyka.

Projekt polegał na stworzeniu gry 2D w stylu „Worms”. Jest to gra na dwóch graczy w której każdy gracz posiada swoją drużynę „robaków” rozmieszczonych na mapie. Celem graczy jest pokonanie wrogiej drużyny poprzez wyeliminowanie wszystkich stworków przeciwnika przy użyciu broni.

2. Narzędzia

- Język programowania: C++

- Biblioteka SFML (Simple and Fast Multimedia Library)

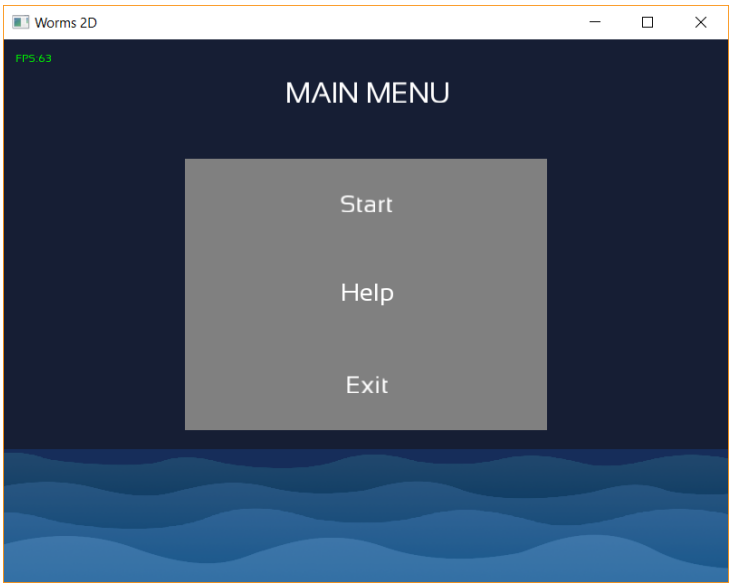
- IDE: Microsoft Visual Studio Community 2017

- OS: Windows 10

3. Działanie i obsługa.

Aby uruchomić projekt należy włączyć plik wykonywalny „*Worms 2D.exe*” lub otworzyć „*Worms 2D.sln*” w programie Visual Studio i skompilować dla architektury x86.

Po uruchomieniu pojawi się następujące okno:

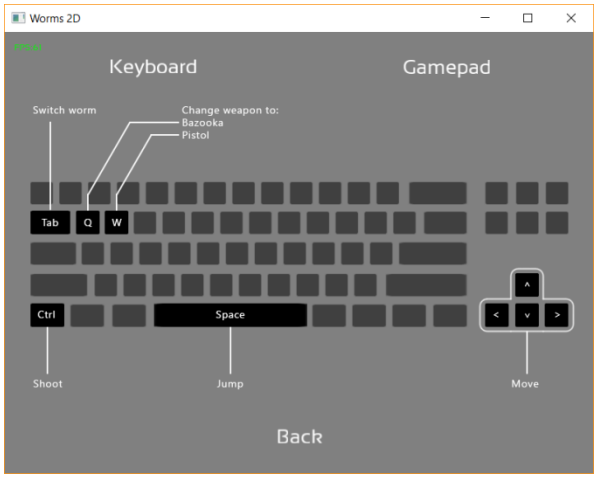


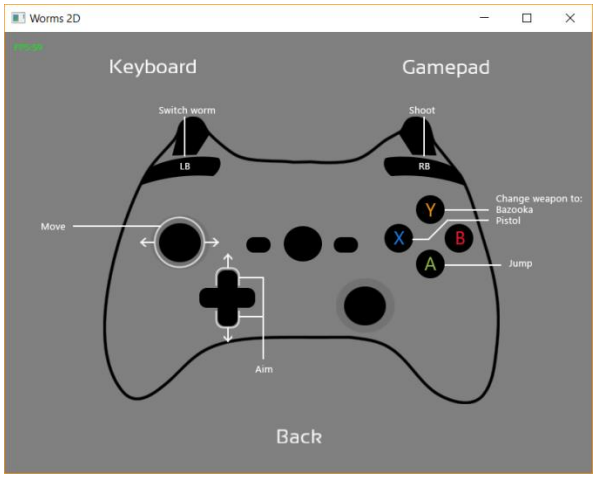
Przyciski w menu obsługiwane są za pomocą myszki.

Poszczególne opcje:

- **Start** – uruchamia nową grę,

- **Help** – pomoc dotycząca sterowania,





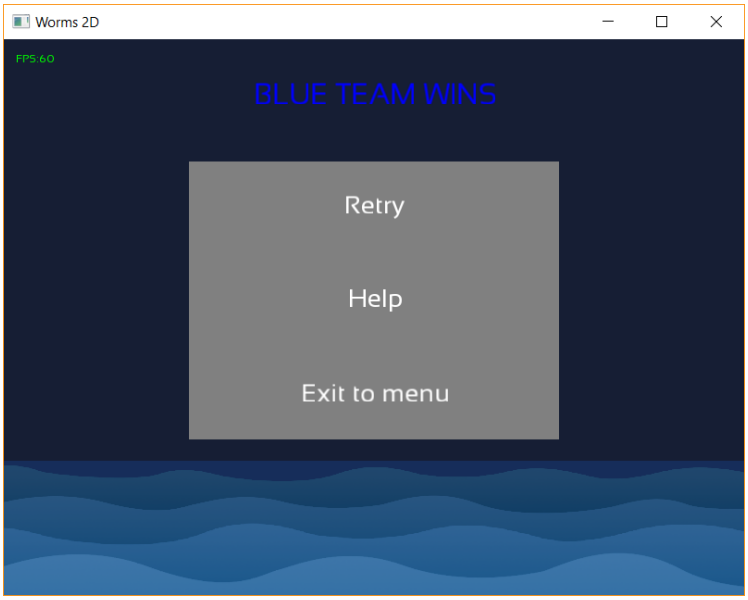
W menu pomocy można przełączać się między wskazówkami dotyczącymi sterowania na klawiaturze lub przy pomocy kontrolera, lub cofnąć się do poprzedniego menu.

- **Exit** – wyjście z gry.

Wyjść z gry lub cofnąć się do poprzedniego menu można za pomocą przycisku ESC.

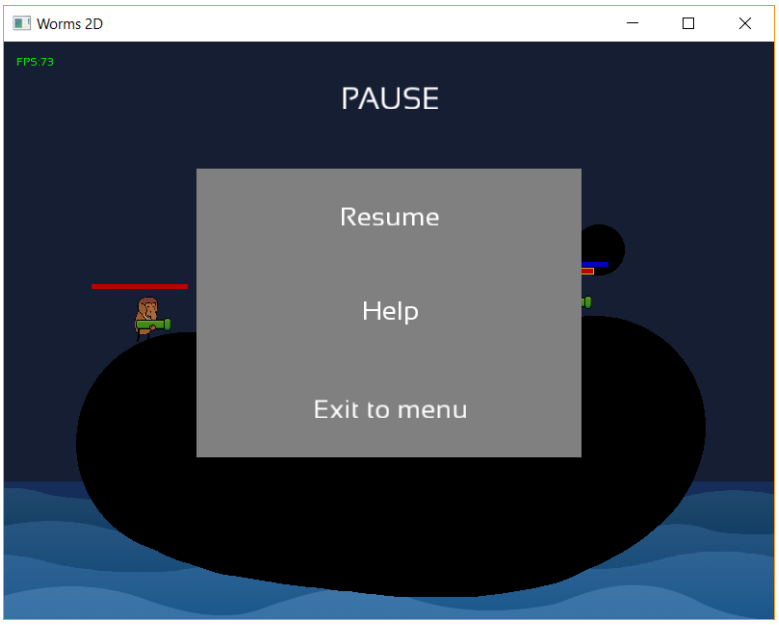
Po wybraniu „Start” na ekranie widzimy po 3 postacie 2 drużyn.

Nad postaciami znajduje się ich pasek życia. Postać która ma podświetlony pasek na żółto jest postacią aktywną. Drużyny ruszają się na przemian. Sterowanie opisane jest w sekcji „Help”. W przypadku wyeliminowania członków drużyny któregoś z graczy wyświetla się odpowiedni ekran, np.



W przypadku wygranej drużyny niebieskiej. Z tego miejsca można przejść do sekcji Help, menu głównego lub zagrać jeszcze raz.

Podczas rozgrywki po naciśnięciu przycisku ESC pojawia się menu pauzy.



- **Resume** – pozwala kontynuować grę.

- **Help** – pozwala zapoznać się ze sterowaniem (działa jak zaprezentowano w sekcji menu głównego).

- **Exit** – powrót do menu głównego.

4. Opis klas metod i funkcji.

Klasa:	**Opis:**
Bullet	Obiekt pocisku. Dziedziczy po klasie SFML Drawable
Metody:	
Konstruktor Bullet(float x, float y)	Tworzy obiekt o pozycji podanej w parametrach
void draw(sf::RenderTarget & target, sf::RenderStates states)	Rysuje obiekt na ekranie. Przeciążona metoda z SFML
float getPosX()	Zwraca X pozycji pocisku na ekranie
float getPosY()	Zwraca Y pozycji pocisku na ekranie
float getScale()	Zwraca wartość skali pocisku
sf::Vector2f getScaleVector()	Zwraca wektor skali pocisku
sf::Sprite getSprite()	Zwraca sprite(SFML) pocisku
sf::Vector2f getVelocity()	Zwraca wektor prędkości pocisku
void setPosX(float x)	Ustawia wartość X pozycji pocisku na ekranie.
void setPosY(float y)	Ustawia wartość Y pozycji pocisku na ekranie.
void setRotation(float rotation)	Ustawia rotację pocisku
void setScale(float scale)	Ustawia skalę pocisku
void setScaleVector(sf::Vector2f scale)	Ustawia wektor skali pocisku.
void setVelocity(sf::Vector2f velocity)	Ustawia wektor prędkości pocisku.
void setVelocity(float x, float y=0)	Ustawia wektor prędkości pocisku
void update	Aktualizuje obiekt pocisku

Klasa:	**Opis:**
Weapon	klasa dziedzicząca po klasie SFML Drawable
Metody:	
Konstruktor Weapon(float x, float y)	Tworzy obiekt. Parametry służą do inicjalizacji pozycji na ekranie.
void draw(sf::RenderTarget & target, sf::RenderStates states)	Rysuje obiekt na ekranie. Przeciążona metoda z biblioteki SFML
Bullet * getBullet()	Zwraca wskaźnik na obiekt pocisku broni
float get Damage()	Zwraca obrażenia(float) danej broni
bool getIsShooting()	Zwraca wartość logiczną – informuje o tym czy broń aktualnie strzela
float getPosX()	Zwraca współrzędną X broni
float getPosY()	Zwraca współrzędną Y broni
float getRotation()	Zwraca rotację broni
float getScale()	Zwraca skalę broni
sf::Vector2f getScaleVector()	Zwraca wektor skali broni
sf::Sprite getSprite()	Zwraca sprite(SFML) broni
void playShootSound()	Odtwarza dźwięk wystrzału broni
void setBullet(Bullet *bullet)	Ustawia pocisk broni na wskazany w parametrze obiekt
void setDamage(float damage)	Ustawia obrażenia od danej broni
void setIsShooting(bool isShooting)	Ustawia stan strzelania broni na podany w parametrze
void setPosX(float x)	Ustawia współrzędną x broni na podaną w parametrze
void setPosY(float y)	Ustawia współrzędną y broni na podaną w parametrze
void setRotation(float rotation)	Metoda ustawiająca rotację broni na podaną w parametrze
void setScale(float scale)	Ustawia skalę broni na podaną w parametrze
void setScaleVector(sf::Vector2f scale)	Ustawia wektor skali broni na podany w parametrze
void shoot(float angle)	Metoda sprawiająca że broń wystrzeli pod kątem podanym w parametrze
void update()	Metoda do aktualizacji obiektu broni

Klasa:	**Opis:**
Bazooka	Klasa dziedzicząca po klasie Weapon.
Metody: Patrz Weapon	

Klasa:	**Opis:**
Revolver	Klasa dziedzicząca po klasie Weapon.
Metody: Patrz Weapon	

Klasa:	**Opis:**
Button	Klasa dziedzicząca po klasie SFML Drawable. Obiekt przycisku do menu.
Metody:	
Konstruktor Button(float position_x,float position_y, const char * label, int id)	Konstruktor przycisku. Parametry służą do ustawienia pozycji x, y, napisu na przycisku,id – funkcja przycisku po kliknięciu (deklaracje w enum game_states)
void clear()	Czyści stan przycisku
void click()	Ustawia stan przycisku na naciśnięty
void draw(sf::RenderTarget & target, sf::RenderStates states)	Rysuje obiekt na ekranie. Przeciążona metoda z biblioteki SFML
int getId()	Zwraca ID przycisku
sf::Vector2f getPosition()	Zwraca wektor pozycji przycisku
sf::Vector2f getSize()	Zwraca rozmiar przycisku
void hover()	Ustawia stan przycisku na podświetlony
bool isClicked()	Zwraca wartość oznaczającą czy przycisk jest kliknięty
bool isHovered()	Zwraca wartość oznaczającą czy przycisk jest podświetlony
void setColorClicked()	Ustawia kolor przycisku na kliknięty
void setColorDefault()	Ustawia kolor przycisku na czysty
void setColorHovered()	Ustawia kolor przycisku na podświetlony

Klasa:	**Opis:**
Menu	Klasa dziedzicząca po klasie SFML Drawable. Menu zawierające przyciski.
Metody:	
Konstruktor Menu()	Tworzy obiekt menu
void changeMenu(int choice)	Zmienia układ przycików i napisów w menu w zależności od wyboru wywołując metody
void clear()	Czyści układ menu.
void clearClickedButton()	Czyści kliknięty przycisk
void draw(sf::RenderTarget & target, sf::RenderStates states)	Rysuje obiekt na ekranie. Przeciążona metoda z biblioteki SFML
Button * getButton(float x, float y)	Zwraca przycisk który znajduje się w określonych współrzędnych

Klasa:	**Opis:**
FPSCounter	Klasa licznika FPS
Metody:	
Konstruktor FPSCounter(unsigned int fontSize = 12)	Tworzy licznik FPS o podanej czcionce lub domyślnej 12.
void FPSCounte::drawFPS()	Rysuje ilość FPS na ekranie
void setColor(sf::Color color)	Ustawia kolor licznika na podany w parametrze
void start()	Uruchamia licznik.

Klasa:	**Opis:**
Terrain	Klasa dziedzicząca po SFML Drawable. Odpowiada za obsługę terenu.
Metody:	
Konstruktor Terrain()	Tworzy obiekt.
void draw(sf::RenderTarget & target, sf::RenderStates states)	Rysuje obiekt na ekranie. Przeciążona metoda z biblioteki SFML
void erase(const sf::Drawable &eraser)	„Rysuje” w terenie kształt przezroczystością – usuwa część terenu.
void reset()	Resetuje mapę do stanu początkowego

Klasa:	**Opis:**
Water	Klasa dziedzicząca po SFML Drawable. Odpowiada za obsługę tła w postaci wody.
Metody:	
Konstruktor Water()	Tworzy obiekt wody.
void draw(sf::RenderTarget & target, sf::RenderStates states)	Rysuje obiekt na ekranie. Przeciążona metoda z biblioteki SFML
void update()	Aktualizuje obiekt wody

Klasa:	**Opis:**
GameEvent	Klasa zdarzeń gry
Metody:	
Konstruktor GameEvent()	Tworzy obiekt.
GameEvent * GetEventInstance()	Zwraca statyczny wskaźnik na obiekt zdarzeń gry (singleton)
sf::Event GetInstance()	Zwraca instancje zdarzeń SFML
void handleEvents()	Obsługuje zdarzenia

Klasa:	**Opis:**
GameSound	Klasa dźwięków gry
Metody:	
Konstruktor GameSound()	Tworzy obiekt
void StartSample()	Odtwarza odgłos worma
void PauseSample()	Pauzuje odgłos worma.
void StopSample()	Zatrzymuje odgłos worma.
void PlayMainMusic()	Odtwarza muzykę w grze
void StartRevolverSound()	Odtwarza odgłos wystrzału rewolwera
void StopRevolverSound()	Zatrzymuje odgłos wystrzału rewolwera
void StartBazookaSound()	Odtwarza odgłos wystrzału bazooki
void StopBazookaSound()	Zatrzymuje odgłos wystrzału bazooki
void PlayDeath()	Odtwarza dźwięk śmierci worma
void StopDeath()	Zatrzymuje dźwięk śmierci worma
void PlayWin()	Odtwarza dźwięk zwycięstwa
void StopWin ()	Zatrzymuje dźwięk zwycięstwa

Klasa:	**Opis:**
GameWindow	Klasa okna głównego.
Metody:	
Konstruktor GameWindow(unsigned int with, unsigned int height, std::string name)	Tworzy obiekt okna głównego o podanej szerokości, wysokości i tytule.
sf::RenderWindow *GetInstance()	Zwraca wskaźnik na instancję okna głównego
void ChangeFrameLimit(unsigned int limit=60)	Zmienia limit klatek na podany w parametrze lub domyślnie 60.
void MainLoop()	Przetwarza główną pętlę okna co klatkę.
void UpdateWorms(int i)	Aktualizuje odpowiedniego worma z wektora 1 drużyny
void UpdateWormsB(int i)	Aktualizuje odpowiedniego worma z wektora 2 drużyny
Worm ** GetCurrentWorm()	Zwraca podwójny wskaźnik na aktualnego worma
std::vector<Worm *>* GetWormsArray()	Zwraca wskaźnik na wektor 1 drużyny
std::vector<Worm *>* GetWormsArrayB()	Zwraca wskaźnik na wektor 2 drużyny
int GetWormCount()	Zwraca wielkość 1 drużyny
int GetWormCountB()	Zwraca wielkość 2 drużyny
int * GetCurrentTeam()	Zwraca wskaźnik na ID aktualnej drużyny
void SetBackgroundColor(sf::Color color)	Ustawia kolor tła
GameSound * GetGameSound()	Zwraca wskaźnik na obiekt dźwięków gry.
void SwitchToRedTeam()	Zmienia aktualną drużynę na Czerwoną (zmiana tury).
void SwitchToBlueTeam()	Ustawia stan aktualnego worma na wybrany a pozostałych na nieaktywny.
static GameWindow * GetGameWindowInstance(Unsigned int width = 1280, unsigned int height=1024, std::string name = „Worms 2D”)	Zwraa wskaźnik na instancję okna głównego lub tworzy je jeśli nie istniało.

Klasa:	**Opis:**
Worm	Klasa dziedzicząca po SFML Drawable. Odpowiada za obsługę postaci
Metody:	
Konstruktor Worm()	Tworzy obiekt worma.
void draw(sf::RenderTarget & target, sf::RenderStates states)	Rysuje obiekt na ekranie. Przeciążona metoda z biblioteki SFML
void update()	Aktualizuje obiekt worma
float left()	Zwraca pozycję lewej krawędzi obiektu worma
float right()	Zwraca pozycję prawej krawędzi obiektu worma
float top()	Zwraca pozycję górnej krawędzi obiektu worma
float bottom()	Zwraca pozycję dolnej krawędzi obiektu worma
float getWormX()	Zwraca współrzędną X obiektu worma
float getWormY()	Zwraca współrzędną Y obiektu worma
void stopMove()	Zatrzymuje worma
void moveLeft()	Porusza worma w lewo
void moveRight()	Porusza worma w prawo
Void jump()	Sprawia że worm skacze
Void damage(float dmg)	Zadaje obrażenia podane w parametrze wormowi
bool isAlive()	Sprawdza czy worm żyje
bool checkCollision(sf::Vector2f point)	Sprawdza kolizję punktu worma z terenem
Int getOffsetY(sf::Vector2f point)	Zwraca offset o jaki ma zostać przesunięty worm aby nie zapadał się w teren
float getScale()	Zwraca skalę worma
sf::Sprite getSprite()	Zwraca obiekt rysowanego worma
void setWeapon(Weapon *weapon)	Ustawia daną broń na taką do której podamy wskaźnik
Weapon *getWeapon()	Zwraca wskaźnik na obiekt aktualnie trzymanej broni
void deleteWeapon()	Usuwa broń którą trzyma worm
void setColMap(sf::Image *image)	Ustawia mapę z którą worm ma mieć kolizję
sf::Text getDebugTxt()	Zwraca tekst dla debug'a
bool isLookingOnLeft()	Sprawdza czy worm patrzy w lewo
bool hasWeapon()	Sprawdza czy worm ma broń
void lookLeft()	Obraca worma w lewo.
void lookRight()	Obraca worma w prawo
void setChoosen()	Ustawia worma jako wybranego do sterowania
void setNormal()	Ustawia worma jako nie do sterowania

5. Praca wykonana przez poszczególnych członków.

Adrian Jakubczyk:

- Grafika, kolizje, menu, teren, sprawozdanie, rotacje broni, sterowanie

Dominik Kaliszewski:

- Dźwięki, bronie, szkielet programu, sterowanie, dokumentacja, koordynowanie kontroli wersji