**Silnik – Jacek Woloniewicz**

Gra jest zrobiona na autorskim silniku "StudentEngine".

Głównych zadaniem silnika jest zarządzanie obiektami typu "GameObject".

Każdy z tych obiektów posiada:

location - obiekt ktory przechowuje pozycje GameObjectu

name - nazwa gameObjectu

group - groupa gameObjectku

image - zdjecie gameObjectku

onEnable() - Wykonywana podczas uruchomienia silnika

onTick() - Wykonywan po każdym obrocie głównej pętli

onCollision() - w przypadku kolizji z innym obiektem "GameObject"

isActive() - Kiedy zwraca False GameObject nie jest rysowany na ekranie

setImage() - Ustawia sprite'a oraz colliderBox'a dla obiektu

getColliderBox() - zwraca obszar kolizji obiektu

destory() - usuwa obiekt przy kolejnym obrocie głównej petli

Aby dodać nowy obiekt GameObject do silnika

należy wywołać metode

studentEngine.addGameObject(gameObject)

>Główna pętla slinka znajduje sie w metodzie "run" jest ona opowiedzialna za:

- sprawdzenie "Eventow" z pygames tzn. czy uzytkonik kliknął klawisz, ruszył myszką, zmienił wiekkosc okna, itd

- wywołanie metody "doTick" we wszystkich obietkach typu "gameObject"

- rysowanie kolejnej klatki za pomocą "doGraphic"

- przechwytywanie wyjątków

> Rysowanie na ekranie znajduje się w metodzie "doGraphic"

- Zarysowuje caly obszar na czarno

- Przygotowuje "Surface"

- Nanosi wszystkie gameObjety ktore znajduje się w zasięgu kamery na "Surface"

- Skaluje "Surface" odpowiednio do zmiennej "zoom\_factor"

- Rysuje "UI" na ekranie

- Rysuje kuror

- na sam koniec za pomocą "pygame.display.update()" wszystko zostaje naniesione na ekran

> Wczytwanie plików

"loadSound(path)" wczytuje pliku typu .mp3/wav z folderu "Resources/Sounds"

"loadSounds(path)" wczytuje wszystkie pliki typu .mp3/wav z folderu "Resources/Sounds"

"loadImage(path)" wczytuje pliku typu .png z folderu "Resources/Textures"

"loadImages(path)" wczytuje wszystkie pliki typu .png z folderu "Resources/Textures"

> Kolizje

"checkCollision(gameObject,targetGroup) sprawdza czy gameObject koliduje z innym gameObjectem ktory nalezy do grupy "targetGroup"

"checkCollisions(gameObject,targetGroup) sprawdza czy gameObject koliduje z innymi gameObjectemi ktore nalezą do grupy "targetGroup"

> Pozostale metody

"findGroup(groupName)" zwraca grupe GameObjectow

"findGameObject(name)" zwraca GameObject o podanej nazwie

"setCamera(x,y)" ustawia pozycje kamerty

"setZoom(factor)" ustawia przyblizenie kamery

"setTitle(name)" ustawia nazwe okna programu

"setCursorImage(image)" ustawia zdjecie kursora

"getScreenSize()" zwraca wielkosc okna programu

"getMouseLocation()" zwraca pozycje myszki na ekranie

**Logowanie – Michał Fiedler**

Uruchamiane jest na początku, przed wejściem do gry. Zrealizowane jest w interfejsie konsolowym.

addUser() – Dodaje nowego użytkownika, przypisuje mu login, hasło i wybraną przez użytkownika postać

findUser() – Sprawdza czy użytkownik o podanym loginie już istnieje

load() – wczytuje plik userdata.json

save() – Zapisuje podane przez użytkownika parametry w celu możliwości ponownego logowania do pliku userdata.json

login() – Wyświetla i pobiera od użytkownika login i hasło, w wypadku gdy użytkownik tworzy konto wywołuje funkcje addUser()

pickcharacter() – Wyświetla dostępne postacie oraz wywołuje funkcje save() przypisując postać do danych logowania użytkownika

**Player – Michał Fiedler**

Odpowiada za stworzenie postaci gracza, kolizje i funkcje.

setUser() –

onEnable() – Funkcja uruchamiana przy włączeniu programu, ustawia wygląd postaci, lokacje, statystyki i kamerę na bohatera

onTick() – Funkcja wykonywana przy każdym obrocie głównej pętli

onCollision() – Sprawdza czy bohater nie wchodzi w kolizje z żdanym obiektem „gameObject”

zoomControl() – Pozwala użytkownikowi na oddalenie kamery

controlEntity() – Umożliwia graczowi na poruszanie postacią

rozmawiaj() – Odtwarza kwestie dialogową

uciekaj() – Postać wykonuje szybkie przemieszczenie się w określoną stronę

atakuj() – Umożliwia graczowi zaatakowanie NPC

przedstawSie() – Odtwarza kwestię bohatera

**NPC – Jacek Woloniewicz**

Odpowiada za zachowania postaci niezależnych.

onEnable() – Funkcja uruchamiana wraz z programem, ustawia obiektowi najważniejsze parametry

onTick() - Funkcja wykonywana przy każdym obrocie głównej pętli, sprawdza kolizje z innymi obiektami gameObject, odpowiada za poruszanie się i mowę NPC

onCollision() – Sprawdza kolizje obiektu i wykonuje odpowiednią interakcje

talking() – Odpowiada za automatyczne odtwarzanie kwestii dialogowych w zależności od odległości od gracza

smoothAudio() – Dostosowuje głośność w zależności od odległości gracza od NPC

walking() – Odpowiada za poruszanie się postaci NPC

**Camera – Jacek Woloniewicz**

Odpowiada za ustawienie kamery. Używa funkcji matematycznej lerp do oddalania i przybliżania kamery. Funkcja lerp powoduje płynny ruch oraz oddalanie. Kamera renderuje gameObjecty w określonym polu widzenia.

onEnable() – Uruchamia się wraz z programem

setTarget() – Ustawia miejsce w jakim znajduje się kamera

setZoom() – Ustawia oddalenie kamery (ustawia zoom())

onTick() - Funkcja wykonywana przy każdym obrocie głównej pętli

zoom() – Oddalenie kamery im większy zoom() tym większy zasięg widzenia

follow() – Odpowiada za poruszanie się kamery za obiektem oznaczonym jako Target

**Water – Michał Fiedler**

Odpowiada za poruszanie się wody na mapie.

Kod, tekstury i mapa zostały stworzone przez Michał Fiedler i Jacek Woloniewicz własnoręcznie.