do uruchomienia gry nalezy przeniesc folder /res/ do folderu z plikiem EXE.

AGameState: public QWidget

Abstrakcyjna klasa reprezentująca stan gry np. MainMenu, Game, Level, Pause itp.
Klasa dziedziczy po Qwidget, dzięki czemu będzie można je wyświetlać w QWindow jako centralWidget. Stany z QGraphicsView będą wtedy wymagać by widok był dzieckiem naszego stanu.
Każdy stan ma swoje ID (Qstring) i funkcje czysto wirtualne

onEnter() - uruchamiana na wejściu stanu

onExit() - uruchamiana na wyjściu stanu

oraz play() - służy do uruchomienia stanu

Narazie AGameState posiada także składowe takie jak QGraphicsScene i QGraphicsView które jak myśle nie będą pożniej potrzebne(niektore stany nie będą ich wymagać).
Klasa posiada również sygnały changeState(AGameState*) i pushState(AGameState*) służące do obsługi managera stanów.

GameStateManager: public QObject

Klasa służąca do zarządzania stanami gry. Posiada publiczne sloty:

pushState(AGameState*) służy do wrzucania stanu na stos i jego obsługe, przy czym stan poprzedni pozostaje na stosie w uśpieniu.

popState() usuwa aktualny stan(zwalnia również pamięć po nim)

changeState(AGameState*) podmienia aktualnie używany stan na stan podany w parametrze

Manager automatycznie łączy sloty z sygnałami stanów. Manager zwalnia pamięć po stanach. Za tworzenie stanów odpowiedzialne są inne stany i przesyłane są do Managera za pomocą sygnałów,

TextureManager

Klasa ta jest singletonem czyli czymś w rodzaju obiektu globalnego.
Dostęp do niej zyskujemy dzięki statycznej funkcji getInstance().

Odpowiedzialna jest za wczytywanie QPixmap (dalej jako tekstura), ich przechowywanie oraz zabezpiecza nas przed skasowaniem używanej tekstury(dzieki std::shared_ptr).

Przy wczytywaniu tekstur podajemy ścieżkę do pliku w odniesieniu do pliku *.exe.
Ścieżka jest również ID tekstury dzięki któremu będziemy mogli dostać shared_ptr<QPixmap> przy użyciu funkcji getTexture(QString ID)

funkcja deleteAll() na razie usuwa wszystkie tekstury, muszę ją zmodyfikować by usuwała tylko nieużywane

AProtype :public QObject

Abstrakcyjna klasa z funkcją wirtualne clone() zwracającą wskaźnik na QGraphicsItem
klasa przyda się dla aktorów tworzonych w dużych ilościach np. przeciwnicy, wizualizacje
pocisków itp(tzw aktorów). Klasa pochodna skonstruowana pod konkretny typ aktora,
odpowiedzialna będzie za wczytanie i przechowywanie shared_ptr z tekstura, dźwięków
i wszystkich tych rzeczy których wczytywanie było by zbyt kosztowne podczas gry oraz zmniejsza
wagę gotowych aktorów.

GraphicsItemFactory : public QOject

fabryka aktorów , przyjmuje wskaznik na AProtype i wywołuje jego unkcje clone(), w
przykładzie użyłem tej funkcji jako karabinu w statku gracza, i spawnera dla przeciwników.