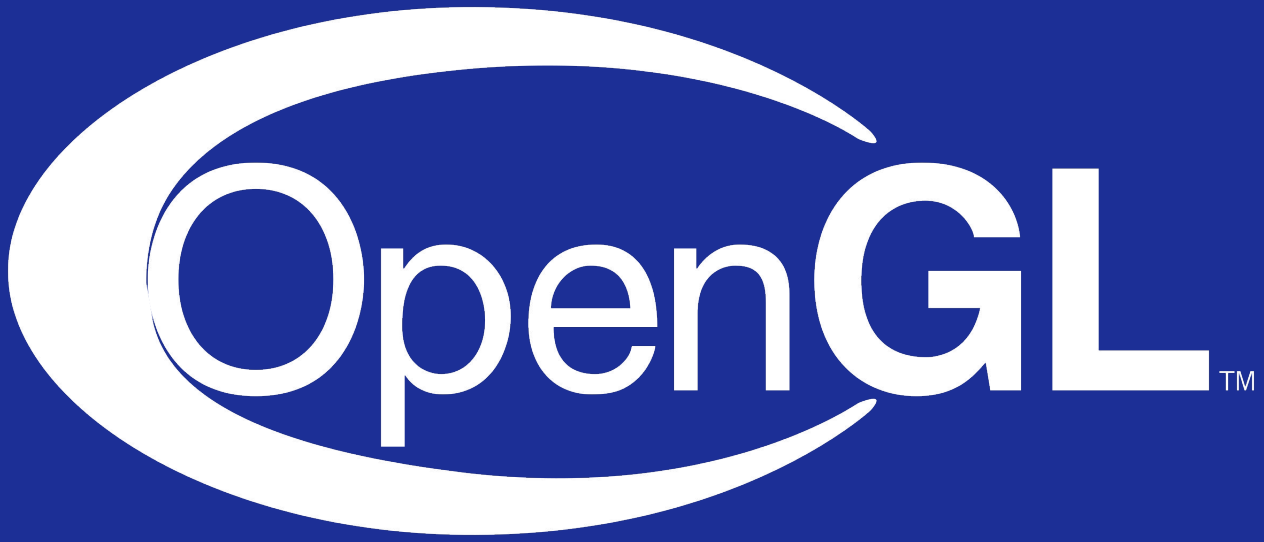
Zachodniopomorski
Uniwersytet Technologiczny
w Szczecinie

Wprowadzenie do QT OpenGL

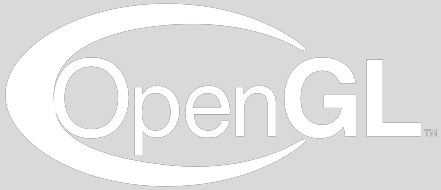


mgr inż. Michał Chwesiuk
mgr inż. Tomasz Sergej
inż. Patryk Piotrowski

Wydział
Informatyki

Zachodniopomorski
Uniwersytet Technologiczny
w Szczecinie

OpenGL - Open Graphics Library



- **Open G**raphics **L**ibrary
- **API** pozwalające na wykorzystanie akceleracji sprzętowej do **renderowania grafiki czasu rzeczywistego**.
- OpenGL **nie jest biblioteką!** Jest to zbiór funkcji i numerycznych stałych, które są interpretowane przez system operacyjny i/lub sterownik karty graficznej.
- Działanie OpenGL można rozumieć jako **maszynę stanu**, raz ustawiona wartość parametru systemu pozostaje taka sama do momentu jej zmiany.
- Główną zaletą jest **wieloplatformowość**.
- Konkurencja: Microsoft **DirectX** i Khronos Group **Vulcan**.
- **Nie obejmuje obsługi wejścia i integracji z systemem operacyjnym w celu utworzenia kontekstu.**

Wydział
Informatyki

Środowisko QT

- **Qt** - zestaw przenośnych bibliotek i narzędzi programistycznych dedykowanych dla języków **C++**, QML i Java.
- Środowisko Qt jest dostępne dla platform: X11 (m.in. GNU/Linux, BSD, Solaris), Windows, Mac OS X, Haiku oraz dla urządzeń wbudowanych opartych na Linuksie (Qt Extended), Windows CE, Symbian, Android.
- Biblioteki Qt zawierają niezależne od platformy systemowej moduły obsługi procesów, plików, sieci, grafiki trójwymiarowej **(OpenGL)**, baz danych (SQL), języka XML, lokalizacji, wielowątkowości, zaawansowanej obsługi napisów oraz wtyczek.
- https://www.qt.io

Zachodniopomorski
Uniwersytet Technologiczny
w Szczecinie

Shadery - programy cieniujące

Wydział
Informatyki

Zachodniopomorski
Uniwersytet Technologiczny
w Szczecinie

Shader - program cieniujący

- **Vertex shader** - odpowiedzialny za wyliczenie pozycji na ekranie.

```
attribute vec4 vertex;
attribute vec3 normal;
uniform mat4 projMatrix;
uniform mat4 mvMatrix;
void main() {\n"
        gl_Position = projMatrix * mvMatrix * vertex;
};
```

- **Fragment shader** - odpowiedzialny za wyliczenie koloru piksela.

```
void main() {
        gl_FragColor = vec4(0.0, 1.0, 0.0, 1.0);
}
```

- **gl_Position** - pozycja na ekranie
- **gl_FragColor** - kolor piksela

Wydział
Informatyki

Zachodniopomorski
Uniwersytet Technologiczny
w Szczecinie

Transformacje macierzy w OpenGL

Wydział
Informatyki

Zachodniopomorski
Uniwersytet Technologiczny
w Szczecinie

Transformacje macierzy w OpenGL

- **Macierz projekcji**

$$\begin{bmatrix} v'_x \\ v'_y \\ v'_z \\ v'_w \end{bmatrix} = \begin{bmatrix} Projection \\ matrix \end{bmatrix} \times \begin{bmatrix} View \\ matrix \end{bmatrix} \times \begin{bmatrix} Model \\ matrix \end{bmatrix} \times \begin{bmatrix} v_x \\ v_y \\ v_z \\ v_w \end{bmatrix}$$

- **Macierz translacji**

glTranslatef(T_x, T_y, T_z) - przesunięcie o wektor [T_x T_y T_z]

$$\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Wydział
Informatyki

Zachodniopomorski
Uniwersytet Technologiczny
w Szczecinie

Transformacje macierzy w OpenGL

- **Macierz obrotu**

glRotatef(ϕ, 1, 0, 0)
- obrót wokół osi X o kąt ϕ podany w stopniach

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\phi & -sin\phi & 0 \\ 0 & sin\phi & cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

glRotatef(ϕ, 0, 1, 0)
- obrót wokół osi Y o kąt ϕ podany w stopniach

$$\begin{bmatrix} cos\phi & 0 & sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -sin\phi & 0 & cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

glRotatef(ϕ, 0, 0, 1)
- obrót wokół osi Z o kąt ϕ podany w stopniach

$$\begin{bmatrix} cos\phi & -sin\phi & 0 & 0 \\ sin\phi & cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Wydział
Informatyki

Transformacje macierzy w OpenGL

- **Macierz skalowania**

glScalef(S_x, S_y, S_z) - przemnożenie o wektor $[S_x\ S_y\ S_z]$

$$\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

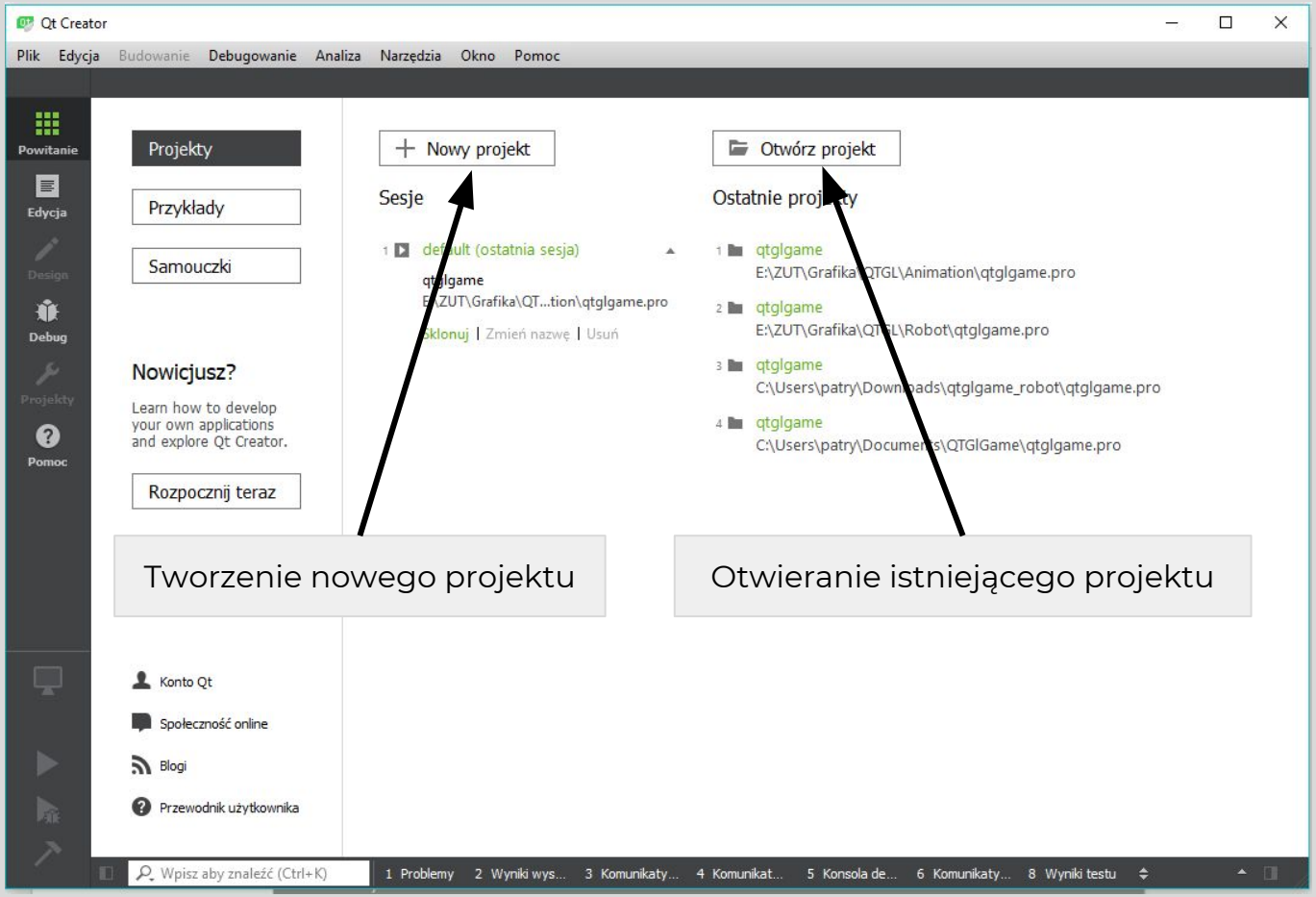
Zachodniopomorski
Uniwersytet Technologiczny
w Szczecinie

Konfiguracja i przygotowanie projektu

Wydział
Informatyki

Zachodniopomorski
Uniwersytet Technologiczny
w Szczecinie

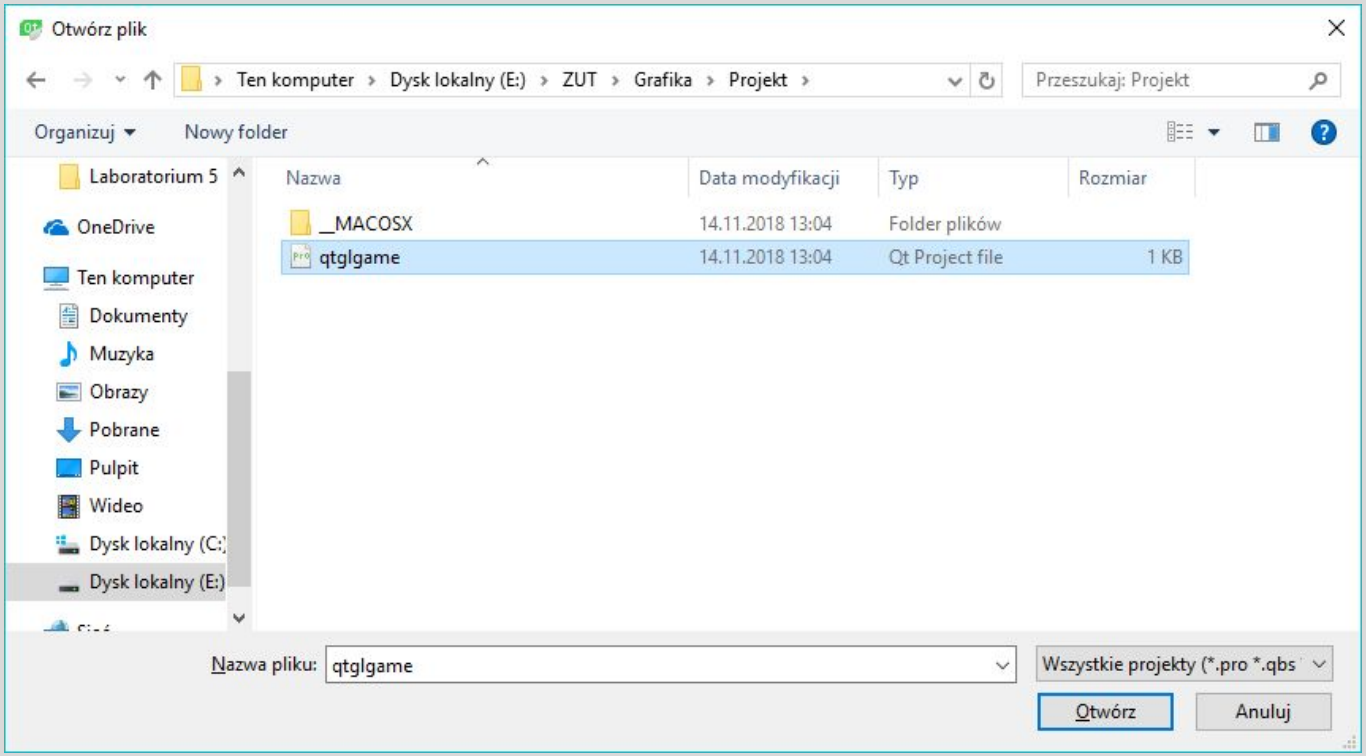
Widok okna QT Creator
Wczytanie projektu



Wydział
Informatyki

Zachodniopomorski
Uniwersytet Technologiczny
w Szczecinie

Widok okna QT Creator
Wczytanie projektu

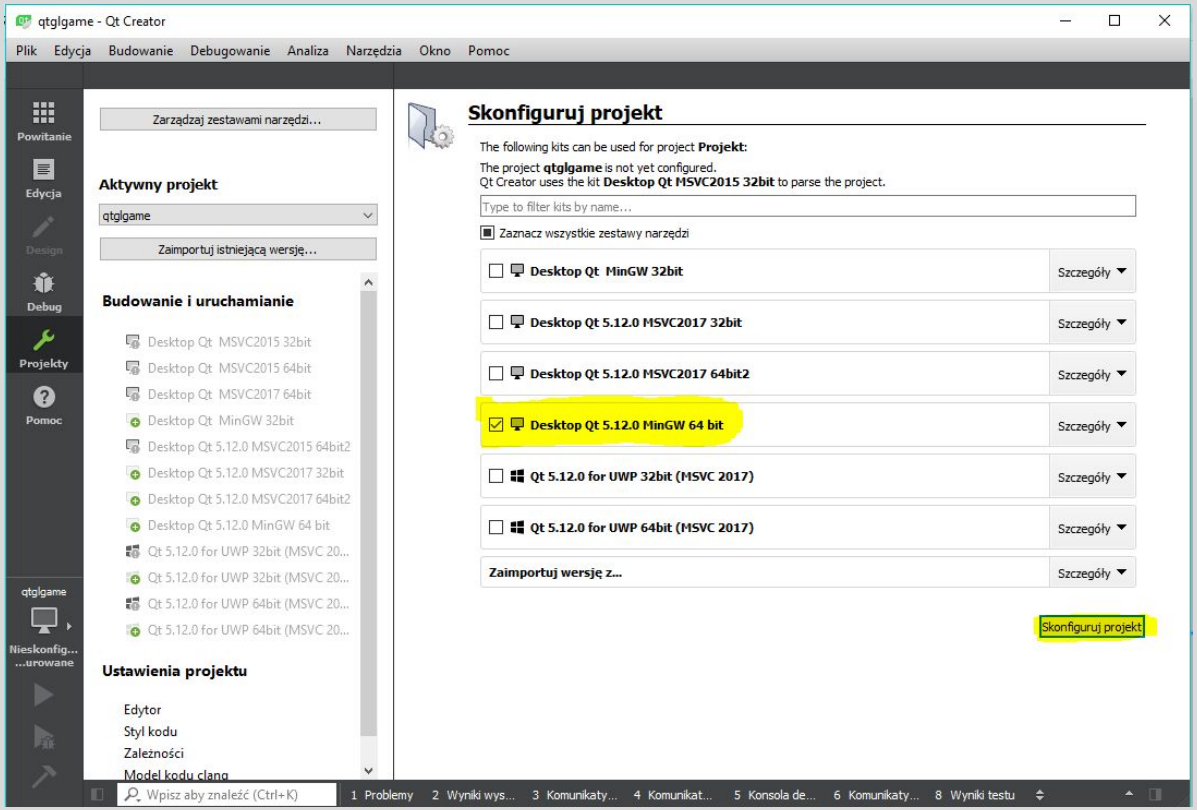


- Otwórz projekt -> wybieramy plik **qlglgame.pro** z wypakowanego projektu

Wydział
Informatyki

Zachodniopomorski
Uniwersytet Technologiczny
w Szczecinie

Widok okna QT Creator
Wczytanie projektu

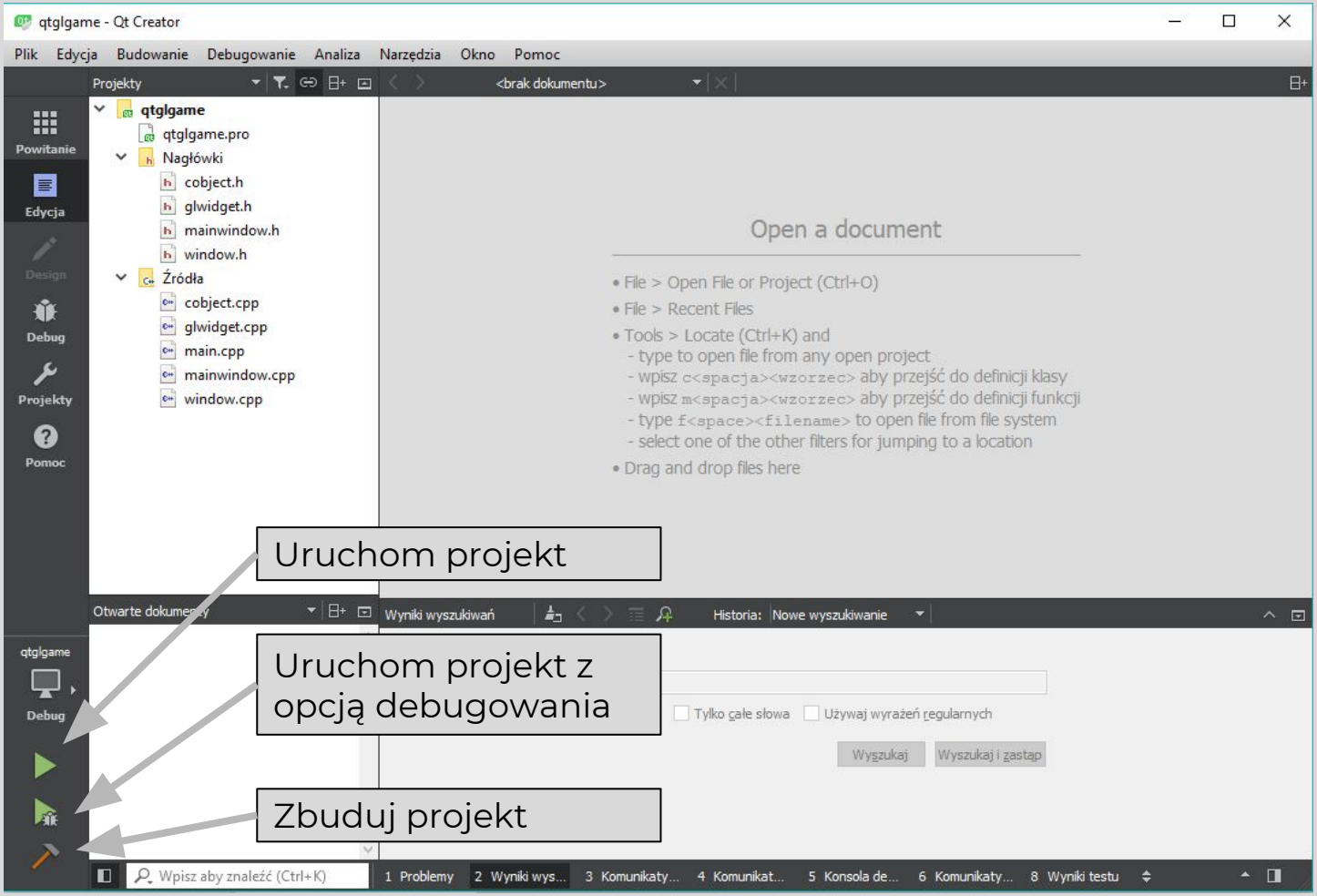


- Następnie należy zaznaczyć odpowiedni zestaw narzędzi *Desktop GT Wersja MinGW 32/64 bit.*
- Potwierdzić wybór przyciskiem *Skonfiguruj projekt*.

Wydział
Informatyki

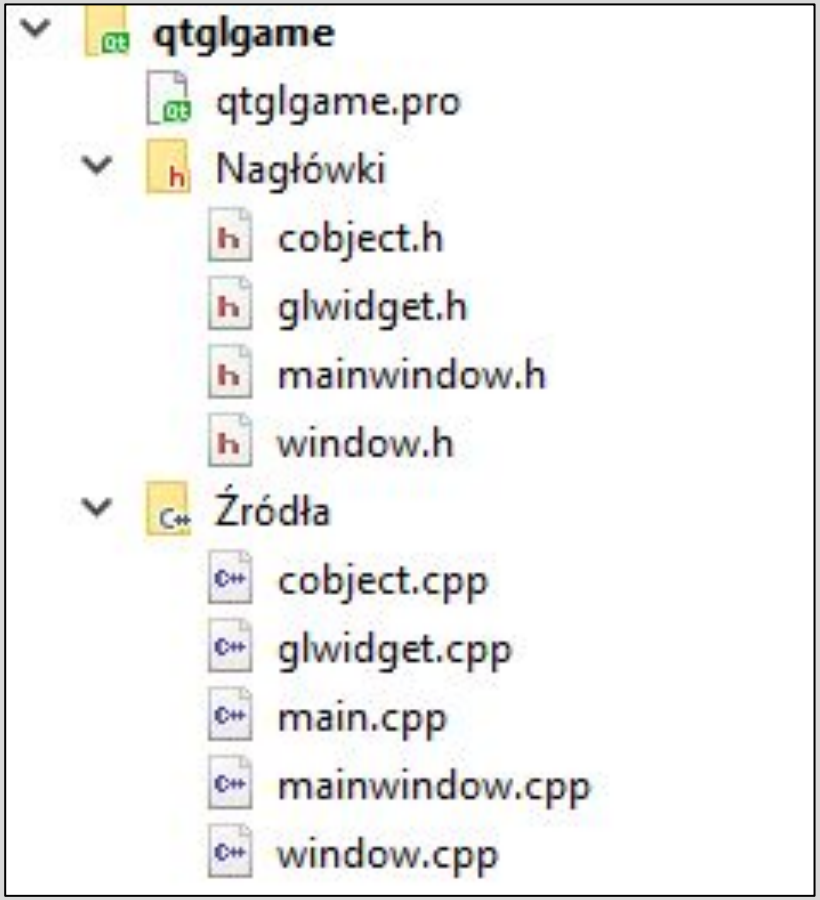
Zachodniopomorski
Uniwersytet Technologiczny
w Szczecinie

Widok okna QT Creator
Widok projektu



Wydział
Informatyki

Widok okna QT Creator
Struktura projektu



Widok okna QT Creator
Klasy

- **CObject** - przechowuje informacje o budowie prymitywów (w projekcie przechowuje informacje o kostce ***(ang. cube)***) - pozycje wierzchołków, ścian oraz wektorów normalnych.
- **MainWindow** - inicjowanie okna kontekstu OpenGL.
- **Window** - okno programu QT OpenGL.
- **GlWidget** - zawiera listę zastosowanych metod, zmiennych, obiektów klas potrzebnych do działania OpenGL - inicjowanie okna, obsługa shaderów, głównej pętli projektu, obsługa urządzeń peryferyjnych (klawiatura, mysz) oraz obsługa zdarzeń *(ang. event)*.

Zachodniopomorski
Uniwersytet Technologiczny
w Szczecinie

Macierz projekcji

Procedura umieszczania obiektu na scenie:

- **m_world = m_world_tmp** - przywracanie domyślnej macierzy projekcji.
- **m_world.scale(QVector3D(1.0, 1.5, 1.2))** - ustawianie skali obiektu QVector3D(względemX, względemY, względemZ).
- **m_world.translate(0.88f, 0.9f, 0.0f);** - zmieniamy pozycje obiektu na scenie - (pozycjaX, pozycjaY, pozycjaZ).
- **m_world.rotate(-65.0f, 0, 0, 1)** - zmieniamy obrót figury na scenie - (ileStopni, dlaOsiX, dlaOsiY, dlaOsiZ).
- **setTransforms();** - akceptujemy wszystkie opcje transformacji obiektu.
- **glDrawArrays(GL_TRIANGLES, 0, m_obj.vertexCount());** - rysowanie figury dla podanych transformacji.
- **Uwaga! Istotna jest kolejność wprowadzania parametrów scale, translate, rotate**.

Wydział
Informatyki

Zachodniopomorski
Uniwersytet Technologiczny
w Szczecinie

Modyfikacja projektu - dodanie obsługi kolorów

Wydział
Informatyki

Zachodniopomorski
Uniwersytet Technologiczny
w Szczecinie

Modyfikacja projektu

Instrukcja opisuje w jaki sposób zmodyfikować kod programu by umożliwić kolorowanie figur w OpenGL.

1. Otwórz plik ***glwidget.cpp*** i odszukaj zmienną *fragmentShaderSource*.
2. Do zmiennej dodaj parametr ***"uniform highp vec3 modelColor;\n"*** przed ciągiem ***"void main() {\n"***.
3. Następnie w tej samej zmiennej odnajdź ciąg **" gl_FragColor = vec4(0.0, 1.0, 0.0, 0.0);\n"** i zamień go na:
   " gl_FragColor = vec4(modelColor, 0);\n"
4. W pliku nagłówkowym ***glwidget.h*** zadeklaruj zmienną ***int m_currentColor***.
5. Po utworzeniu zmiennej udajemy się z powrotem do ***glwidget.cpp*** i odszukujemy funkcję void GLWidget::initializeGL().
6. Odszukujemy linię (124):
   m_mvMatrixLoc = m_program->uniformLocation("mvMatrix");
   i doklejamy poniżej niej:
   m_currentColor = m_program->uniformLocation("modelColor");

- Po wykonaniu powyższych kroków kolory figur możemy zmieniać wywołując następującą procedurę:
  m_program->setUniformValue(m_currentColor, QVector3D(1.0f, 1.0, 1.0));
- Powyższą funkcję wykonujemy każdorazowo przed rysowaniem figury, kolory można ustawiać w zmiennej ***QVector3D(RedColor, GreenColor, BlueColor)*** - w przedziale 0.0f do 1.0f.

Wydział
Informatyki

Zadanie

Zadanie

Wersja podstawowa - 0.75pkt

- Zbuduj robota z wykorzystując cube'a - skorzystaj z opcji skalowania, rotacji i translacji.
- Odpowiednio pokoloruj figury.

Wersja rozszerzona - 1.00pkt

- Wersja podstawowa + dodanie animacji machania ręką. Animacja ta powinna być uzależniona od zmiennej zliczającą klatki na sekundę.