Programowania w OpenGL z użyciem shader'ów, SFML obsługa klawiatury i myszki

1. Cel ćwiczenia:

Zapoznanie z programowaniem grafiki przy użyciu shader'ów, SFML obsługa zdarzeń klawiatury i myszki.

2. Wstęp:

Biblioteka SFML umożliwia obsługę urządzeń peryferyjnych takich jak klawiatura, myszka czy joystick. Obsługa zdarzeń jest możliwa poprzez klasę ***sf::Event***, gdzie przechowywane są wszystkie informacje o zdarzeniach systemowych, które właśnie się wydarzyły.

Zdarzenia są pobierane za pomocą funkcji ***sf::Window::pollEvent*** lub ***sf::Window::waitEvent***.

Instancja ***sf::Event*** zawiera typ zdarzenia (przesunięcie myszy, naciśnięcie klawisza, zamknięcie okna, ...), a także szczegóły dotyczące tego zdarzenia. Należy zwrócić uwagę, że parametry zdarzenia są zdefiniowane w unii, dlatego tylko członek pasujący do typu zdarzenia zostanie poprawnie wypełniony. Wszyscy pozostali członkowie będą mieć niezdefiniowane wartości.

Na przykład, dla zdarzenia ***KeyPressed***, należy odczytać ***event.key,*** wszystkie inne elementy będą niezdefiniowane.

Przykład sprawdzenia zdarzenia naciśnięcia klawisza ESC w celu zamknięcia programu:

```
case sf::Event::KeyPressed:
switch (windowEvent.key.code)
{
case sf::Keyboard::Escape:
running = false;
break;
}
```

W analogiczny sposób można sprawdzić naciśnięcie dowolnego klawisza. Przykład sprawdzenia wciśnięcia klawisza "1" (w tym samym switch'u co wcześniej):

```
case sf::Keyboard::Num1: .....
```

Podobnie można sprawdzić aktualne współrzędne kursora myszy.

W switch ***(windowEvent.type) {*** należy użyć:

case sf::Event::MouseMoved:

if (windowEvent.mouseMove.y > mouse_y) { .....

3. Ćwiczenie:

Należy przygotować projekt umożliwiający rysowanie dowolnego wielokąta foremnego gdzie ruch myszki w pionie powoduje zmianę ilości wierzchołków, co jest na bieżąco odrysowywane (ilość wierzchołków dowolna). Dodatkowo przyciski numeryczne 1,2...0 umożliwiają zmianę zastosowanego prymitywu.

Wskazówki:

W celu zmiany ilości wierzchołków należy zastosować dynamiczą tablicę przechowującą dane zamiast dotychczasowej statycznej.

GLfloat *vertices = new GLfloat[punkty_ * 6];

Ważna informacja: rozmiar przesyłanych danych do pamięci karty graficznej należy określić np. za pomocą funkcji:

glBindBuffer(GL_ARRAY_BUFFER, buffer);

glBufferData(GL_ARRAY_BUFFER, sizeof(GLfloat)*punkty_ * 6, vertices,

GL_STATIC_DRAW);

Metoda określająca typ zastosowanego prymitywu to:

// Narysowanie wielokąta na podstawie wierzchołków

glDrawArrays(prymityw, 0, punkty);

gdzie prymityw może przyjmować wartości:

GL_POINTS, GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP, GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUADS, GL_QUAD_STRIP, GL_POLYGON.

Przykład działania aplikacji:

początkowo ruch myszy w pionie powoduje zmianę liczby punktów na okręgu, później zmiana prymitywów za pomocą klawiszy 1...0