POLITECHNIKA WROCŁAWSKA
Instytut Informatyki, Automatyki i Robotyki
Zakład Systemów Komputerowych

Wprowadzenie do grafiki komputerowej

Kurs: INEK00012L

Sprawozdanie z ćwiczenia nr 2

TEMAT ĆWICZENIA

OpenGL - podstawy

Wykonał:	**Karol Pastewski 252798**
Termin:	**WT/TP 7.30-10.30**
Data wykonania ćwiczenia:	**12.10.2021r.**
Data oddania sprawozdania:	**19.10.2021r.**
Ocena:	

Uwagi prowadzącego:

1. Wstęp teoretyczny

1.1. Dywan Sierpińskiego

Dywan Sierpińskiego to fraktal utworzony z podzielenia kwadratu na dziewięć identycznych kwadratów. Następnie usuwany jest środkowy kwadrat i proces jest powtarzany w nieskończoność. Wyjątkowe w uzyskanym fraktalu jest to, że jego pole powierzchni wynosi zero. Podczas projektowania programu wyświetlającego Dywan Sierpińskiego należy właśnie tę własność uwzględnić, czyli nasz fraktal przy pewnym kroku wykonywania się algorytmu powinien „zniknąć”.

2. Nowe polecenia OpenGL

- `void glBegin(GLenum mode)` [...] `glEnd()` – procedura potrzebna do wyświetlenia na ekranie obiektu pierwotnego. Przykładem obiektu pierwotnego jest kwadrat (`GL_POLYGON`) lub trójkąt (`GL_TRIANGLES`). pomiędzy dwoma wypisanymi funkcjami należy użyć funkcji `glVertex` oraz opcjonalnie `glColor`.
- `void glVertex2f(GLfloat x, GLfloat y)` – komenda używana wewnątrz `glBegin` i `glEnd`, aby określić współrzędne wierzchołka. Aby narysować na ekranie kwadrat należy określić cztery wierzchołki, dla trójkąta trzy wierzchołki itd. Komenda może przyjmować różne parametry zależnie od sufiksów, które użyliśmy. W tym przypadku „2” oznacza dwa parametry, a „f” oznacza typ parametrów `GLfloat`.
- `void glColor3f(GLfloat red, GLfloat green, GLfloat blue)` – komenda ustawiająca obecny kolor. Tak samo jak poprzednia komenda parametry mogą się zmienić zależnie od użytego sufiksu. Aby rysowany obiekt miał jednolity kolor należy tylko raz wypisać tą komendę, ale można także przed każdą komendą `glVertex2f` zmienić kolor na nowy, aby otrzymać efekt gradientu (co zostało użyte w programie).

3. Rozwiązanie zadania

Szkielet programu oraz funkcja `ChangeSize` nie zostały zmodyfikowane z instrukcji do laboratorium, więc nie będę ich tu opisywał.

```
void RenderScene(void) {

	system("cls");
	glClear(GL_COLOR_BUFFER_BIT);

	int n, def, color;
	int a = 729;

	cout << "Dywan Sierpinskiego" << endl;
	cout << "Dlugosc poczatkowego kwadratu:             729 px" << endl;
	cout << "Krok, w ktorym figura zmiknie z obrazu:    7" << endl << endl;

	do {
		cout << "Ile krokow ma wykonac algorytm? (minimum to 1)" << endl;
		cin >> n;
	} while (n < 1);

	do {
		cout << "Czy dywan ma byc zdeformowany? (0 - nie; 1 - tak)?" << endl;
		cin >> def;
	} while (def < 0 || def > 1);

	do {
		cout << "Czy dywan ma byc kolorowy? (0 - nie; 1 - tak)?" << endl;
		cin >> color;
	} while (color < 0 || color > 1);

	srand(time(NULL));
	initializeCarpet(a, n, def, color);
	cout << endl << "Algorytm sie wykonal";


	glFlush();
	// …

}
```

Różnice zaczynają się w funkcji `RenderScene`, w której przeprowadzane są inicjalizacje potrzebnych zmiennych takich jak długość boku, liczba kroków wykonania algorytmu oraz czy dywan ma być zdeformowany i/lub kolorowy. Oprócz długości boku wszystkie te parametry wpisuje użytkownika, a długość została ustawiona na 729, czyli potęgę 3 tak, aby algorytm prawidłowo pokazał dywan, czyli po 7 kroku dywan powinien „zniknąć” (bo $729 / 3^7 < 1$, więc skoro długość boku jest mniejsza od 1 piksela, to kwadrat się nie narysuje).

```
void initializeCarpet(int a, int n, int deformation, int color) {
	float x1 = a / (-2.0f);
	float y1 = a / (2.0f);

	divideRect(x1, y1, a / 3, n, deformation, color);
}
```

Ta funkcja jedynie oblicza współrzędne dla górnego lewego wierzchołka oraz wywołuje funkcje `divideRect`, która jest funkcją rekurencyjną.

```
void divideRect(float x1, float y1, int a, int n, int deformation, int color) {
	if (n == 0 || a < 1) return;
	n--;

	for (int i = 0; i < 3; i++) {
		for (int j = 0; j < 3; j++) {
			if (i == 1 && j == 1) continue;

			float newX1 = x1 + (i * a);
			float newY1 = y1 - (j * a);

			divideRect(newX1, newY1, a / 3, n, deformation, color);

			if (n == 0) {
				createRect(newX1, newY1, a, deformation, color);
			}
		}
	}
}
```

Funkcja wywołuje rekurencyjnie samą siebie dopóki nie dojdzie do ostatniego kroku albo długość boku będzie mniejsza od 1 (nie ma sensu kontynuować algorytmu skoro kwadrat i tak nie będzie wyświetlony na ekranie). Jeżeli algorytm przejdzie do ostatniego kroku to dopiero wtedy tworzy kwadraty na ekranie przy funkcji `createRect`.

```
void setColor(int color) {
	if (color == 1) {
		glColor3f((rand() % 101) / 100.0, (rand() % 101) / 100.0, (rand() % 101) /
100.0);
	}
	else {
		glColor3f(1.0f, 1.0f, 1.0f);
	}
}

void setCoordinate(int a, float& x, float& y) {

	float rangeOfError = 0.20f;
	// …

	float error = a * rangeOfError;
	int intErrorTimes1000 = (int)(error * 1000);
	if (intErrorTimes1000 != 0) {
		x = x + (rand() % (2 * intErrorTimes1000) - intErrorTimes1000) / 1000.0f;
		y = y + (rand() % (2 * intErrorTimes1000) - intErrorTimes1000) / 1000.0f;
	}
}

void createRect(float x, float y, int a, int deformation, int color) {

	glBegin(GL_POLYGON);
	setColor(color);
	if (deformation == 1) {
		setCoordinate(a, x, y);
	}
	glVertex2f(x, y);
	setColor(color);
	glVertex2f(x + a, y);
	setColor(color);
	glVertex2f(x + a, y - a);
	setColor(color);
	glVertex2f(x, y - a);
	glEnd();
}
```

Funkcja `createRect` używa dwóch pomocniczych funkcji `setColor` oraz `setCoordinate`. `setColor` kiedy zostanie wybrana opcja kolorowego Dywanu będzie losowała liczbę w zakresie od 0 do 1 z dokładnością do dwóch miejsc po przecinku. Jeżeli nie będzie wybrana opcja koloru, to wypełnienie jest białe. `setCoordinate` oblicza przesunięcie współrzędnych kwadratu. Wielkość przesunięcia jest określana przez zmienną `rangeOfError`, który jest procentowy i oblicza maksymalne przesunięcie, które określa zmienna `error`. Obliczać resztę z dzielenia można tylko przez liczbę stałoprzecinkową dlatego występowanie `intErrorTimes1000`, aby znaleźć liczbę z zakresu od `<-intErrorTimes1000,intErrorTimes1000>`, a później podzielić ją przez 1000 i otrzymać prawidłowy zakres `<-error,error>`.