Opis kodu programu RadioSimulator

1. Tryb Draw scene

W trybie draw scene jedyne znaczące funkcje ze względu na radiokomunikację to:

Konwersja dBm na Watty:

$$P[Watts] = \frac{10^{\frac{P[dBm]}{10}}}{1000}$$

Skalowanie układu współrzędnych

$$x_{scaled} = x \cdot SCALE$$
$$y_{scaled} = y \cdot SCALE$$

Kwantowanie punktów na układzie współrzędnych:

$$x_q = x \cdot \left| \frac{x}{grid_x} + 0.5 \right|_{floor}$$
$$y_q = y \cdot \left| \frac{y}{grid_y} + 0.5 \right|_{floor}$$

2. Tryb single ray

Ten tryb korzysta z metod **propagate** i **get_dist_coef_array** obiektu promienia, które są zdefiniowane w klasie **Ray** w pliku ray.py

Metoda **propagate**, dopóki parametr AP promienia jest większy od 0, oblicza punkty przecięcia ze wszystkimi ścianami na scenie. Następnie filtruje wyniki ze względu na:

- Czy obliczony punkt fizycznie znajduje się na ścianie
- Czy obliczony punkt znajduje się w pożądanym kierunku

Odfiltrowane wyniki są sortowane względem odległości rosnąco i wybierany jest najbliższy punkt przecięcia. Następnie nowy wektor kierunkowy promienia jest obliczany funkcją **reflection_vec** z pliku geometrics.py, a parametr AP jest zmniejszany o 1.

Gdy żaden punkt przecięcia nie przejdzie procesu filtracji, program oblicza przecięcie z granicami sceny i tam kończy propagację promienia.

Metoda **get_dist_coef_array** oblicza współczynnik dystansu dany wzorem:

$$coef = \frac{\alpha}{d} \cdot e^{2j \cdot \pi \cdot \frac{fd}{c}}$$

Dla kolejnych punktów na trasie propagacji promienia z krokiem step. Gdy program wykryje że nowe koordynaty do obliczeń są poza danym segmentem trasy promienia, współczynnik odbicia alpha jest

mnożony przez ten sam współczynnik ściany, od której następuje odbicie, oraz zostaje wyznaczony nowy zakres pracy do kolejnego odbicia.

Moc w tym trybie jest liczona z wzoru:

$$P = P_{ref} \cdot |coef|^2, gdzie$$
$$P_{ref} = P_{TX} \cdot \left(\frac{\lambda}{4\pi}\right)^2$$

Ponieważ zakładamy zysk anten = 1 w skali liniowej, nie zostały one uwzględnione we wzorach.

Pomimo, że współczynnik dystansu nie był w tym wypadku potrzeby, został użyty dla spójności z resztą trybów.

3. Tryb multi ray

Tryb multi ray korzysta z metod **propagate_to_point** oraz **get_coef_at_end,** które są zdefiniowane w klasie Ray w pliku rays.py.

Metoda **propagate_to_point** propaguje promień z nadajnika do odbiornika, odbijając go od zadanych ścian. Obliczanie ścieżki promienia jest za pomocą odbić lustrzanych względem ścian, od których promień ma się odbić. Odbicie jest realizowane funkcją **point_mirror_line** z pliku geometrics.py

Następnie liczony jest punkt przecięcia linii promienia ze ścianą i obliczany jest wektor odbity analogicznie to trybu single ray.

Metoda na ten moment nie uwzględnia potencjalnych przeszkód, które mogą wystąpić na trasie promienia, dlatego proszę o szczególną troskę podczas używania tego trybu.

Metoda **get_coef_at_end** działa w analogiczny sposób do metody używanej w trybie single ray. Zamiast obliczać wektor współczynników, od razu oblicza wypadkowy współczynnik odbicia oraz dystans i zwraca jeden współczynnik.

Te dwie metody są wywoływane dla każdego narysowanego promienia. Następnie wypadkowa moc jest obliczana z wzoru:

$$P = P_{ref} \cdot \left|\sum_i coef_i\right|^2$$
$$P_{ref} = P_{TX} \cdot \left(\frac{\lambda}{4\pi}\right)^2$$

4. Tryb diffraction

Tryb diffraction służy do symulacji dyfrakcji metodą Deygout. Tryb korzysta z metod **get_diffraction**, **get_diffraction_power** oraz **distance_spaces.**

Metoda **get_diffraction_power** oblicza i zwraca współczynnik dystansu(ten sam co w poprzednich trybach) oraz dodatkowe tłumienie wynikające z możliwej dyfrakcji(w dB). Funkcja sprawdza punkty przecięcia promienia skierowanego bezpośrednio z nadajnika do odbiornika z wszystkimi ścianami na scenie. Następnie filtruje uzyskane punkty, ze względu na kierunek promienia i czy znajdują się na

danej ścianie (metoda analogiczna do metody **propagate**). Jeżeli jakieś punkty przeszły proces filtracji, następnie są porównywane dystanse pomiędzy nadajnikiem i odbiornikiem oraz nadajnikiem i punktem przecięcia. Jeżeli obliczony punkt przecięcia jest bliżej niż odbiornik oznacza to że odbiornik nie jest w LOS.

Gdy odbiornik jest w LOS współczynnik dystansu jest obliczany identycznie to trybu multi ray, a tłumienie wynikające z dyfrakcji wynosi 0(ważne w kontekście wykresu w skali dB). Następnie moc jest obliczana analogicznie to trybu multi ray z użyciem mocy odniesienia P_{ref}.

Gdy odbiornik nie jest w LOS, współczynnik dystansu dalej jest obliczany jak w przypadku LOS, jednak dodatkowo jest obliczane tłumienie wynikające z dyfrakcji. Obliczanie są odległości pomiędzy nadajnikiem i punktem dyfrakcji(d1), punktem dyfrakcji i odbiornikiem(d2) oraz odległość punktu dyfrakcji od prostej definiowanej przez punkty nadajnika i odbiornika. Wykorzystywane są tutaj funkcje **point_point_distance** i **point_line_distance.**

Następnie obliczany jest współczynnik v:

$$v = h \cdot \sqrt{\frac{2}{\lambda} \cdot \left(\frac{1}{d_1} + \frac{1}{d_2}\right)}$$

Tłumienie jest obliczane ze wzoru:

$$c[dB] = 6.9 + 20\log_{10}\left(\sqrt{(v-0.1)^2+1} + v - 0.1\right)$$

Funkcja **get_diffraction** jest wrapperem dla metody **get_diffraction_power**. Gdy flaga mode jest równa True, funkcja zwraca samo tłumienie wynikające z dyfrakcji. Gdy flaga jest równa False, tłumienie jest konwertowane na skale liniową i uwzględniane przy współczynniku:

$$c_{linear} = 10^{-\frac{c}{10}}$$
$$coef_{new} = coef * c_{linear}$$

5. Obliczanie współczynnika odbicia

Metoda obliczania współczynnika odbicia jest zawarta w klasie **Wall** w pliku props.py. Bazuje ona na strukturze danych materiałów. Gdy parametr **custom_alpha** jest równy True, zostanie zwrócona wartość alpha zapisana w materiale. W przeciwnym wypadu zostanie obliczy zgodnie ze uproszczonymi wzorami z normy ITU-R P.2040-2.

$$R_{eTE} = \frac{\cos\theta - \sqrt{\eta - \sin^2\theta}}{\cos\theta + \sqrt{\eta - \sin^2\theta}}$$

$$R_{eTM} = \frac{\eta\cos\theta - \sqrt{\eta - \sin^2\theta}}{\eta\cos\theta + \sqrt{\eta - \sin^2\theta}}$$

$$T_{eTE} = \frac{2\cos\theta}{\cos\theta + \sqrt{\eta - \sin^2\theta}}$$

$$T_{eTM} = \frac{2\sqrt{\eta}\cos\theta}{\eta\cos\theta + \sqrt{\eta - \sin^2\theta}}$$

Ze względu na dużą liczbę problemów technicznych, funkcja została zaimplementowana na chwilę przed terminem końcowym i nie została dokładnie przetestowana. Dlatego w przypadku problemów zalecane jest ustawienie flagi **custom_alpha** na True i użycie stałego współczynnika odbicia

6. Krótki opis zawartości plików

Files.py – zawiera metody zapisu i odczytu sceny pliku .json

Geometrics.py – zawiera podstawowe funkcje geometryczne wykorzystywane przez inne funkcje

Globals.py – zawiera zmienne globalne i konfiguracyjne programu. Tam można zmienić skalę czy tryb obliczania współczynnika odbicia.

Main.py – główny plik programu, to on powinien być uruchamiany

Materials.py – lista predefiniowanych materiałów, gdy zapisana scena posiada inne zostaną one wczytane do programu, jednak nie będzie można ich używać do tworzenia innych ścian

Props.py – zawiera definicję klasy reprezentującej ścianę, transmitter, receiver, materiał

Ray.py – zawiera definicję klasy reprezentującej promień i funkcje pomocnicze z nim związane

Routines.py – zawiera funkcje obsługi zdarzeń głównej pętli programu, analiza zaleca z autorem

Window.py – zawiera definicje układu GUI

Pliki.json – zawierają gotowe układy sceny