Laboratorium programistyczne — Python: Trójwymiarowy świat z sześcianów

Projekt „Matematyka dla Ciekawych Świata”,
Robert Ryszard Paciorek
<rrp@opcode.eu.org>

2020-05-11

1 Grafika sześcianów

Koncepcja budowy trójwymiarowego świata z sześcianów udekorowanych prostą teksturą spopularyzowana został ponad 10 lat temu przez grę *Minecraft*.

Na sześciany takie można patrzeć jako na woksele[1] tworzące bardziej złożone obiekty 3D. Podejście takie jest odmienne od najczęściej stosowanego w grafice 3D opartej na siatkach wielokątów, gdzie każdy obiekt opisywany jest siatką najczęściej trójkątów przybliżająca kształt jego powierzchni.

Tworzenie tego typu grafiki polega na umieszczaniu w punktach przestrzeni 3D sześcianów. W klasycznym (wokselowym) podejściu do takiej grafiki sześciany te mają bok o długości 1 i są umieszczane na siatce opisywanej liczbami całkowitymi, w taki sposób ze w danym punkcie siatki jest dokładnie jeden sześcian lub nie ma żadnego. Możliwe jest rozszerzenie tej koncepcji pozwalające umieszczać sześciany na współrzędnych niecałkowitych i doprowadzać do ich nakładania się.

2 Uruchamiamy pierwszy program

Na zajęciach będziemy korzystać z lekko zmodyfikowanej wersji pythonowego klona *Minecrafta*. W tym celu należy utworzyć fork repl'a https://repl.it/@RPaciorek/Craft albo (jeżeli nie korzystasz z repl.it'a) pobrać archiwum http://ciekawi.icm.edu.pl/materialy/edycja_XI/craft.zip i rozpakować jego zawartość. Przygotowany repl, jak i archiwum zawierają 3 pliki:

- `main.py` (z kodem który będziemy modyfikować),
- `craft.py` (z kodem bibliotecznym, który jest nam potrzebny, ale na razie nie będziemy w niego wnikać) oraz
- `texture.png` (z teksturami używanymi do kolorowania naszych sześcianów).

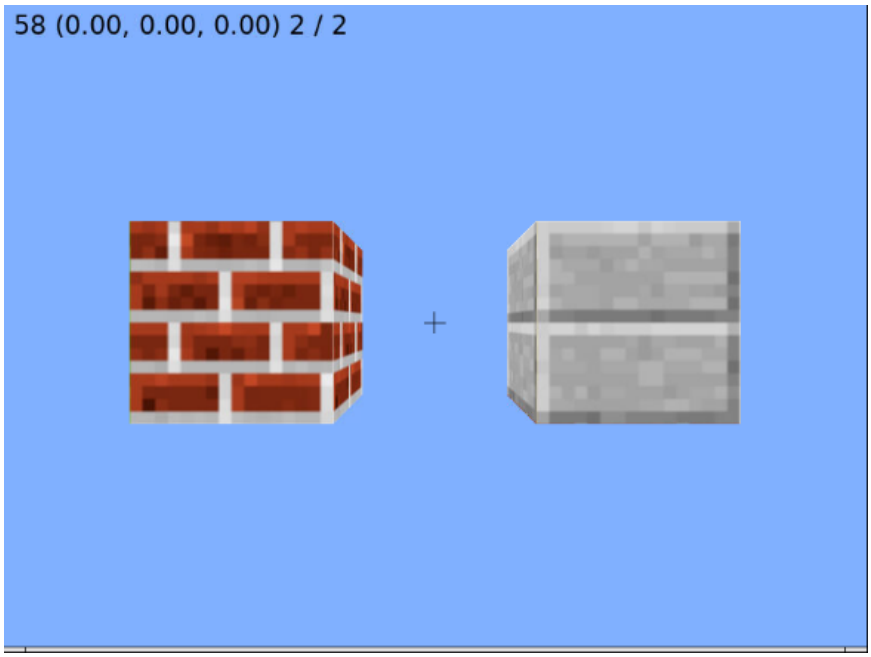
W wyniku uruchamiania naszych programów mogą pojawić się kolejne pliki, ale nie będą one nas interesować. Plik na którym będziemy pracować (`main.py`) ma postać:

```
import pyglet
# potrzebne dla repl.it

from craft import Model, run
model = Model()

# tutaj dodajemy bloki
model.add_block((-1,0,-3), "BRICK")
model.add_block(( 1,0,-3), "STONE")

run(model)
```

1. *voxel* – *volumetric picture element*, najmniejszy element przestrzeni w grafice trójwymiarowej. Można na niego patrzeć jak na trójwymiarowy odpowiednik pixela – pixele są efektem podziału płaszczyzny przy pomocy dwuwymiarowej siatki na części o jednakowych rozmiarach (najczęściej kwadraty), voxele są elementem podziału przestrzeni trójwymiarowej przy pomocy trójwymiarowej siatki na części o jednakowych rozmiarach (najczęściej sześciany).

Na początku importujemy wymagane elementy z `craft.py` oraz tworzymy obiekt `model`, który będzie używany do tworzenie wyświetlanego świata 3D. Następnie tworzymy dwa bloki (sześciany) z różnymi teksturami. Na końcu uruchamiamy aplikację wraz z utworzonym modelem.

Spróbujmy zatem uruchomić aktualną wersję klikając *Run* w repl.it lub uruchamiając plik `main.py` poprzez `python3 main.py`. Powinniśmy zobaczyć okienko z dwoma sześcianami takie jak na poniższej ilustracji:



2.1 Dodawanie bloków

Jak już zauważyliśmy do dodawania bloków używana jest metoda `add_block()` wywoływana na `model`. Przyjmuje ona dwa argumenty:

- pierwszy określa położenie bloku i jest krotką (niemodyfikowalną listą) złożoną z 3 współrzędnych:
  - x – rośnie od lewej do prawej strony ekranu
  - y – rośnie od dołu do góry ekranu
  - z – maleje wgłąb ekranu
- drugi określa teksturę (czyli grafikę która zostanie nałożona na nasz sześcian), mamy 4 możliwości: `"BRICK"`, `"STONE"`, `"GRASS"` i `"SAND"`

2.2 Poruszanie się po świecie

Przy pomocy myszy wskazujemy kierunek w krórym patrzymy. Przy pomocy klawiszy WSAD możemy poruszać się zwględem tego kierunku (w replit wymaga to kliknięcia w okienko z grafiką).

Klikając lewym przyciskiem myszy możemy usunąć wskazany blok (z wyjątkiem bloków `"STONE"`). Klikając prawym przyciskiem myszy możemy dobudować blok od strony wskazanej ściany innego bloku. Typ dobudowywanego bloku ustala się wcześniej przy pomocy klawiszy: 1 (`"BRICK"`), 2 (`"GRASS"`) i 3 (`"SAND"`).

2.3 Zaawansowana interakcja

Modyfikując elementy zdefiniowane w `craft.py` możemy np. przypisywać różne akcje związane z wieloma blokami (np. równoczesne niszczenie wielu bloków, czy dodawanie predefiniowanych zestawów bloków) pod naciśnięcia przycisków lub kliknięcia myszą. Warto w tym celu przyjrzeć się metodom `on_mouse_press` i `on_key_press` zawartym w klasie `Window`.

3 Zadania

Zadanie 3.0.1

Zbuduj wieżę z bloczków o wysokości 13 bloczków. Użyj pętli.

Zadanie 3.0.2

Zmodyfikuj rozwiązanie zadania 3.0.1 tak aby wieża składała się naprzemiennie z bloczków typu `"BRICK"` i `"STONE"`.
Wskazówka: możesz użyć instrukcji warunkowej `if` *i sprawdzania reszty z dzielenia przez dwa.*

Zadanie 3.0.3

Zmodyfikuj rozwiązanie zadania 3.0.2 tak aby wieża składała się naprzemiennie z bloczków typu `"BRICK"`, `"STONE"` i `"GRASS"`.

Zadanie 3.0.4

Napisz funkcję `wieza`, która będzie budowała wieżę o podanej (w jej argumencie) wysokości. Użyj jej do narysowania wieży o wysokości 7 bloczków.
Wskazówka: zmodyfikuj rozwiązanie zadania 3.0.1

Zadanie 3.0.5

Napisz funkcję, która będzie tworzyła wypełniony kwadrat z bloczków o podanej (w jej argumencie) długości boku. Użyj jej do narysowania kwadratu o długości boku 4.
Wskazówka: użyj dwóch pętli jedna wewnątrz drugiej.

Zadanie 3.0.6

Zmodyfikuj rozwiązanie zadania 3.0.5 tak aby powstał kwadrat bez wypełnienia (ramka o grubości jednego bloczka).
Wskazówka: możesz użyć instrukcji warunkowej `if`.

Zadanie 3.0.7

Zmodyfikuj rozwiązanie zadania 3.0.5 tak aby funkcja przyjmowała dwa argumenty i tworzyła prostokąt o podanych długościach boków.

Zadanie 3.0.8

Napisz funkcję, która będzie tworzyła koło z bloczków o podanym (w jej argumencie) promieniu.
Wskazówka: Koło można uzyskać rysując kwadrat w którym pomijamy punkty leżące dalej niż promień od środka koła. Dokonując tego porównania warto mieć na uwadze że nasz bloczek ma bok długości jeden – warto zmniejszyć promień o połowę długości jego boku (czyli 0.5).

Zadanie 3.0.9

Napisz funkcję, która będzie tworzyła sześcian z bloczków o podanej (w jej argumencie) długości boku.

Zadanie 3.0.10

Napisz funkcję, która będzie tworzyła piramidę (ostrosłup o podstawie kwadratu) o podanej wysokości i długości boku podstawy około 2 razy wiekszej.

Zadanie 3.0.11

Zmodyfikuj rozwiązanie zadania 3.0.10 tak aby można było określać niezależnie wysokość i długość podstawy ostrosłupa.
Wskazówka: do konwersji liczb zmiennoprzecinkowych na całkowite możesz użyć funkcji `round`.