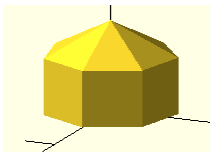
Poznajemy OpenSCAD
VI - powielanie

1. Uruchom program - > nowy projekt

A – projekt chatek

2. Tworzymy taką oto, murzyńską, 7-ścienną chatkę (patrz lekcja III):



3. Jeśli już działa, zróbmy z tych dwóch cylindrów **unię,** żeby było elegancko.

```
union()
]{
    cylinder (5,5,5,$fn=7);
    translate([0,0,5]) cylinder(3,5,0,$fn=7);

}
```

4. Podmieńmy **unię** na **moduł nazwa.**

```
1  module chatka()
2 {
3      cylinder (5,5,5,$fn=7);
4      translate([0,0,5]) cylinder(3,5,0,$fn=7);
5
6  }
```

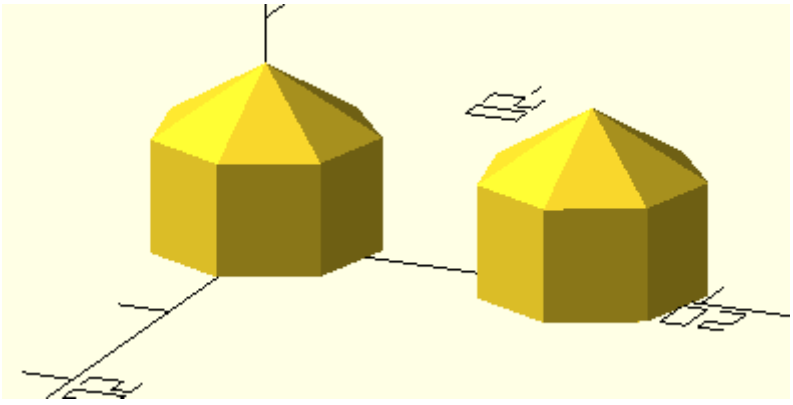
Ojej! Zniknęło!
Tak, bo moduł to tylko projekt. Żeby go zbudować, trzeba wywołać ten projekt:

```
6  }
7
8  chatka();
```

Ponownie widzimy chatkę, ale jako zaprojektowaną **chatka()**

5. Zbudujmy, według tego samego projektu/modułu drugą chatkę:

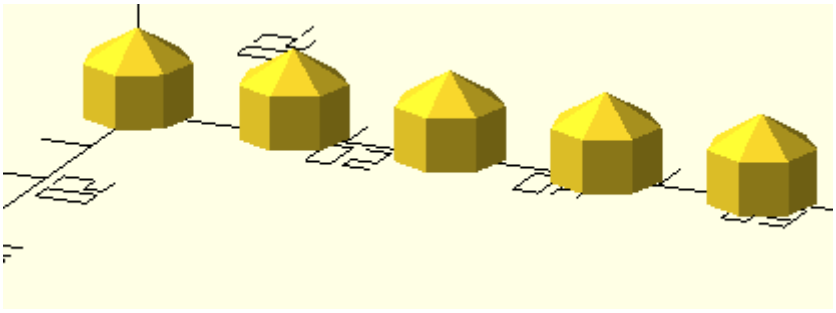
```
9  translate([0,15,0]) chatka();
```



B – projekt chatek przy drodze

Dwie już mamy… a jeśli ma być 20? Albo 120? Informatyk musi to zautomatyzować.

1. Teraz matematyka dla 5 chatek



Pierwsza chatka	przesunięcie = 0	-
Druga chatka	przesunięcie = 15	-
Trzecia chatka	przesunięcie = 30	-
Czwarta chatka	przesunięcie = 45	-
Piąta chatka	przesunięcie = 60	-

Od razu widać, że nasze przesunięcie trzeba wymnożyć **15*nr**

przesunięcie = 0	nr=0	bo 15 * 0 = 0
przesunięcie = 15	nr=1	bo 15 * 1 = 15
przesunięcie = 30	nr=2	bo 15 * 2 = 30
przesunięcie = 45	nr=3	bo 15 * 3 = 45
przesunięcie = 60	nr=4	bo 15 * 4 = 60

I tak właśnie zrobimy:

```
translate([0,nr*15,0]) chatka();
```

tylko **nr** musi być po kolei: **0,1,2,3,4**

Matematycznie się zgadza?

2. To „lecimy”:

dla nr od 0 do 4 przesuń chatki o 15 * nr

```
8  for ( nr=[0:4] )
9      translate([0,nr*15,0]) chatka();
```

tylko tak trochę informatycznie zapisane.

Wyszło?
A co się zmieni jeśli zapiszemy nr=[1:5] ?
A jeśli nr = [-4,4] ?

3*) Da się użyć for wewnątrz for-a… Dasz radę zrobić osiedle 5x5?