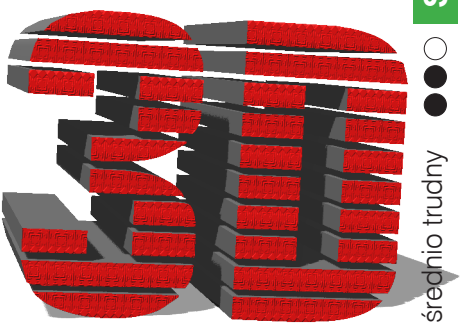
Tym razem poznamy najważniejsze reguły niezbędne do korzystania z jednej z wersji oprogramowania typu CAD. Nauczymy się, jak tworzyć odpowiednie komendy, a na koniec zaprojektujemy podstawowe akcesorium sędziego sportowego!

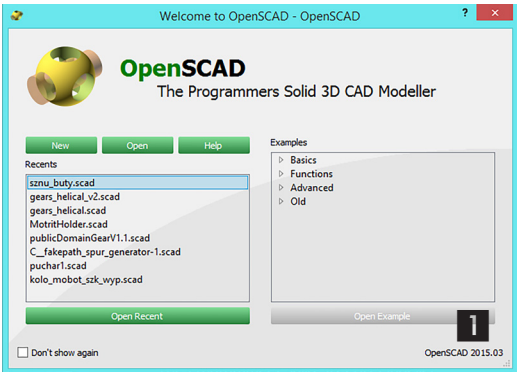
PRAKTYCZNY KURS DRUKU



Lekcja 5 – OpenSCAD, pierwsze kroki

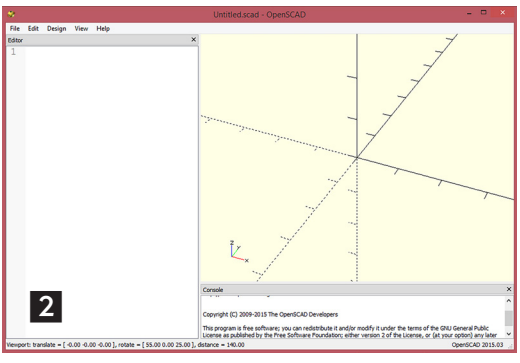
Na rynku znajduje się wiele wersji oprogramowania typu CAD. W ramach naszego kursu przedstawimy program w wersji *freeware*, dostępny na platformę Windows, Linuxy i Mac OSX, ale także na system Android. Mowa oczywiście o oprogramowaniu OpenSCAD, które można pobrać ze strony **www.openscad.org/downloads.html**.

Szybka instalacja i pierwsze uruchomienie – otwiera się okno wyboru projektów, nad którymi ostatnio trwały prace (**1**). Podczas pierwszego uruchomienia historia będzie pusta, należy więc kliknąć przycisk „New” i otworzy się nowy projekt. Można także obejrzeć przykłady, są one dostępne w prawej części okna „Examples”. Pogrupowano je pod kątem stopnia trudności, począwszy od podstawowych (*Basics*) po zaawansowane (*Advanced*).



Po otwarciu nowego projektu należy zapoznać się z interfejsem aplikacji (**2**). Składa się on z trzech okien oraz jednego menu. Okno górne, prawe, to okno prezentacji, będzie można w nim obejrzeć projekt w wersji roboczej, a także finalnej – istnieje wiele trybów wyświetlania projektu, w zależności od potrzeb. Okno prawe dolne to konsola komunikatów, wyświetlane są w nim zarówno błędy, jak i raporty z zakończonych procesów. Okno lewe to miejsce, w którym będzie powstawał kod projektu, o czym za chwilę. Powyżej tego okna znajduje się menu, składające się z pięciu pozycji.

Pozycja pierwsza to **„File”**. Zawiera standardowe opcje służące do otwierania, zapisywania i zamykania projektów. Dość ważnym elementem z punktu widzenia przyszłego drukowania jest funkcja „Export”. Umożliwia zapisanie poprawnie wygenerowanego



projektu w jednym z formatów obsługiwanych przez program do cięcia (*Slicer*). Obecnie najpopularniejszym formatem pozostaje STL, który jest obsługiwany przez każdy program do cięcia.

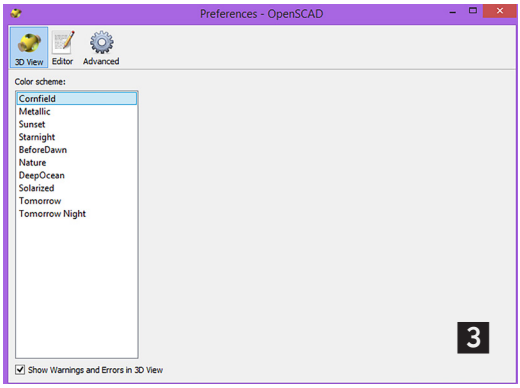
W drugiej pozycji – **„Edit"** – także przewidziano standardowe opcje edycji. Na samym dole znajduje się opcja „Preferences", która umożliwia konfigurację OpenScad (**3**). Zakładka pierwsza to jedynie wybór trybu wyświetlania, domyślnie ustawiono tu tryb „Cornfield", chyba najlepszy. Kolejna zakładka (**4**) służy do konfiguracji edytora. Można tu ustawić rodzaj i wielkość czcionki, rodzaj edytora (domyślnie Qscintilla), kolory funkcyjne (ułatwiają wprowadzanie komend), możliwość ustawienia wcięć, zawijania linii. Na samym dole znajduje się opcja „Enable brace matching", która odpowiada za dopasowywanie nawiasów (podświetlanie otwierającego i zamykającego). Jest ona bardzo pomocna w trakcie tworzenia projektu i ważne, aby była aktywna. Ostatnia zakładka, „Advanced" (**5**), pozwala na ustalenie możliwości wyświetlania ostrzeżeń, aktywowania biblioteki OpenGL1.x, określenie maksymalnej liczby brył dla pojedynczego projektu i wielkości plików *cache* (można zwiększać w przypadku problemów z poprawnym renderingiem), włączanie obsługi wielu projektów jednocześnie, wyłączanie okna powitalnego oraz uaktywnianie możliwości lokalizacji interfejsu (obecnie w trakcie realizacji).

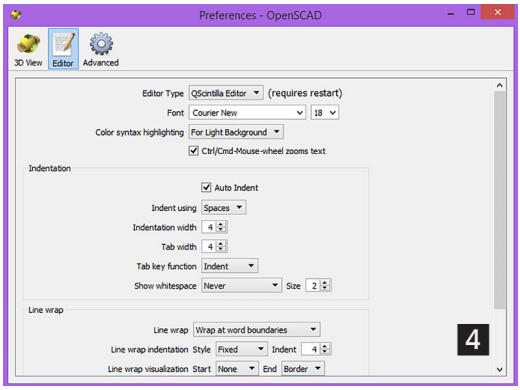
Trzecia pozycja górnego menu – **„Design"** – ma kilka opcji do odświeżania i prezentowania widoku projektu. Najważniejsza z nich to „Preview", dla której przypisano skrót klawiszem F5, oraz „Render" – klawiszem F6. Pierwsza służy do generowania szybkiego podglądu projektu, natomiast druga do dokładnego generowania całego projektu, a następnie eksportowania go jako STL. Jeśli w kodzie znajdą się trudne do zlokalizowania błędy, można wyświetlić bardziej przejrzysty kod z pominięciem komentarzy. Służy do tego funkcja „Display AST...". Komenda „Display CSG Tree" dodatkowo pokazuje drzewo komend.

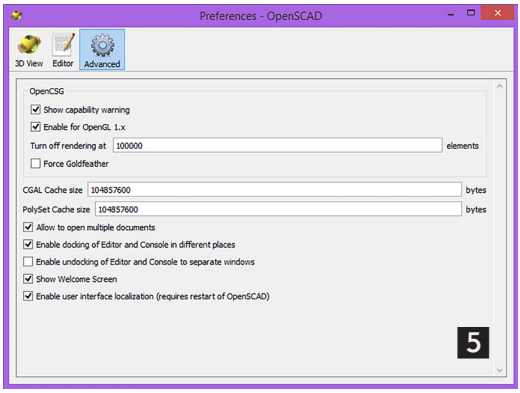
Czwarta pozycja górnego menu – **„View"** – zawiera cały szereg komend służących do poprawnego wyświetlania projektowanych obiektów, skróty klawiszowe do łatwego obracania i zmiany perspektywy. Bardzo przydatne są funkcje „Show Axes" i „Show Scale Markers". Pierwsza włącza osie układu współrzędnych w oknie prezentacji projektu, natomiast druga odpowiada za wyświetlanie na każdej z osi skali liczbowej. Na samym dole pojawiają się jeszcze trzy ważne opcje: „Hide Toolbars" (odpowiada za wyświetlanie menu graficznych), „Hide editor" (za wyświetlanie okna edytora) i „Hide console" (za wyświetlanie okna z komunikatami). Zaleca się, aby wszystkie trzy opcje były nieaktywne.

Zapisujemy komendę

Nadszedł wreszcie czas na utworzenie pierwszego projektu. W normalnym programie byłoby to hasło



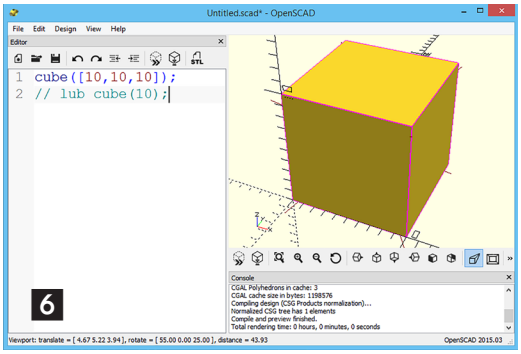




„Hallo Word", w naszym przypadku będzie to popularna figura – sześcian. Służy do tego komenda: "cube([10,10,10]);" (**6**).

Jak widać, składa się ona z polecenia „cube", a następnie podanych w nawiasie parametrów i wartości – jeżeli chodzi o wartości w osi XYZ, muszą być zawsze podane w nawiasach kwadratowych. W tym przypadku w osiach X, Y i Z mamy po 10 mm. Zmieniając wartości w każdej z osi, można uzyskać różnego rodzaju prostopadłościany.

Podstawową jednostkę w naszej strefie językowej stanowią milimetry, więc domyślna konfiguracja OpenSCAD odbywa się także **w milimetrach**



(podczas instalacji dana ta jest pobierana z systemu operacyjnego). W trakcie tworzenia projektu można po kropce używać ułamkowych części mm, jednak należy pamiętać o ograniczeniach samej drukarki. Może się zdarzyć, że wydruk będzie odbiegał od projektu, co staje się szczególnie ważne dla części spasowanych. Wyobraźmy sobie np., że tworzymy dwa cylindry, z których jeden ma wchodzić w drugi, pierwszy ma mieć średnicę wewnętrzną 20,2 mm, drugi zewnętrzną 19,8 mm, a konstrukcja przewiduje luźne wsuwanie cylindrów, co ma umożliwić zachowanie 0,3 mm luzu. Niestety, drukarka ma głowicę o średnicy dyszy 0,5 mm, czyli nie jest w stanie wydrukować elementów o wymiarach 0,2 i 0,8 mm, a jedynie wielokrotność 0,5 mm. W tym przypadku wydruk będzie wynosił po 20 mm dla obu średnic, bez uwzględnienia 0,3 mm luzu, który zostanie pominięty.

Projektując pod konkretny model drukarki, należy brać pod uwagę jej **rozdzielczość druku**. W poziomie wynosi on średnicę dyszy (0,2; 0,3; 0,35; 0,4; 0,5), natomiast w pionie nawet najsłabszy model drukarki potrafi drukować z warstwą 0,1 mm. Średnica dyszy w drukarkach wielkogabarytowych wynosi przeważnie 0,8 mm, a nawet 1,2 mm.

Wracając do naszego projektu… Ponieważ sześcian ma podane trzy wymiary, dlatego zostały użyte nawiasy kwadratowe, można jednak uprościć komendę i podać jeden wymiar, identyczny dla wszystkich ścian. Najlepiej jednak na początek użyć pełnej składni, czyli wspomnianej: ”cube([10,10,10]);” (**7**).

W każdym przypadku zakończenie komendy oznacza się jako „;”, a za średnikiem zaczyna się kolejna komenda. Każda komenda może się składać z wielu poleceń, a w niektórych przypadkach kilka komend można łączyć w jedną. Polecenie może zawierać zmienne, dla których musi być respektowana reguła, zgodnie z którą zmienne dla trzech wymiarów muszą być podawane w nawiasach kwadratowych „[ ]”. Przyjęto przy tym następującą kolejność osi: wartość pierwsza to X, druga to Y, a trzecia to Z, czyli [X,Y,Z].

Komenda na ekranie

Komenda napisana, czas więc wyświetlić ją na ekranie. Służy do tego pierwsza ikona *toolbars* lub klawisz F5. W oknie podglądu powinien pojawić się sześcian lub komunikat o błędzie w oknie komunikatów. Najczęstszym błędem jest **brak poprawnych nawiasów otwierająco-zamykających** lub **brak średnika na końcu komendy**. Czasem zdarza się także pisać polecenia z dużych liter, co jest również klasyfikowane jako błąd. Można go jednak dostrzec już na etapie pisania, ponieważ błędnie stworzone polecenie będzie miało kolor czarny zamiast niebieski (w domyślnej kolorystyce).

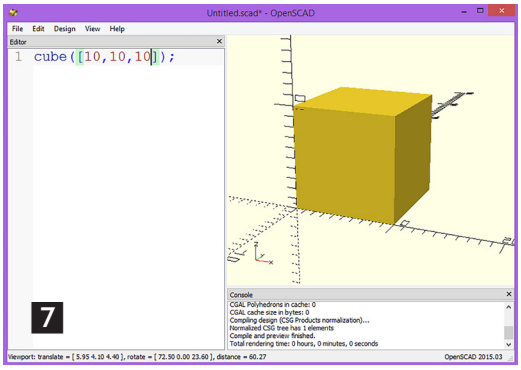
Zakładając, że podgląd się wygenerował poprawnie, na końcu linii warto spróbować użyć komentarza, np. //sześcian.

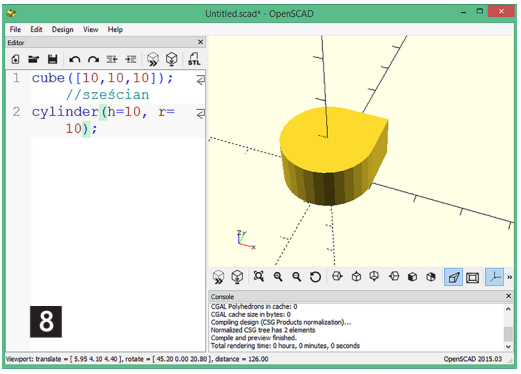
Rysujemy bryłę

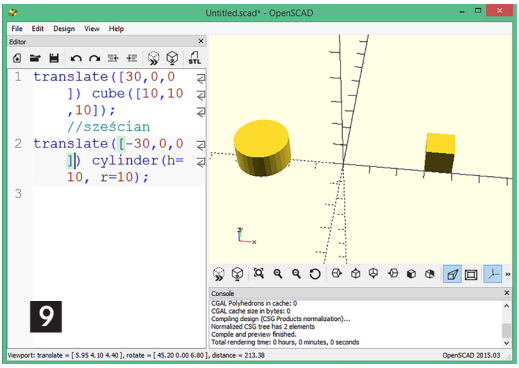
Nadszedł czas na kolejną komendę: rysowanie bryły walca. Do tego celu należy użyć polecenia: ”cylinder(h=10, r=10);”.

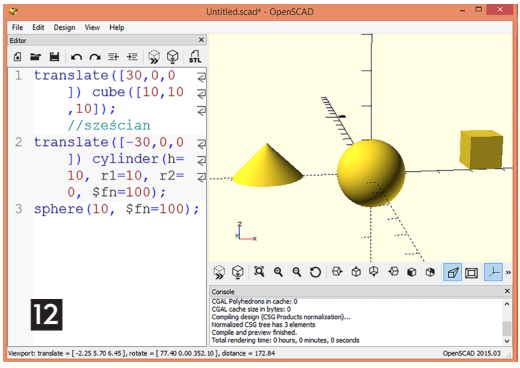
W tym przypadku wartości są podane w nawiasach zwykłych, ponieważ zaznacza się tam tylko wysokość i promień podstawy (**8**). Jak widać na obrazku, bryły się wzajemnie przenikają, tworząc wspólnie obiekt. Warto teraz zapisać obecny stan projektu pod nazwą „projekt1.scad”, naciskając przyciski „Ctrl+S”.

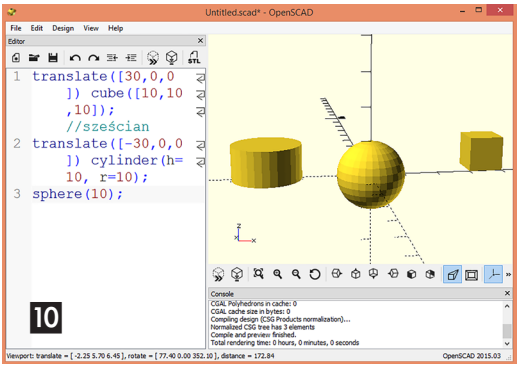
Poszczególne bryły można przesuwać względem początku układy współrzędnych – służy do tego

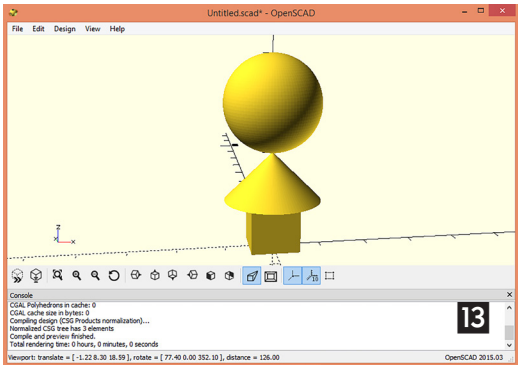


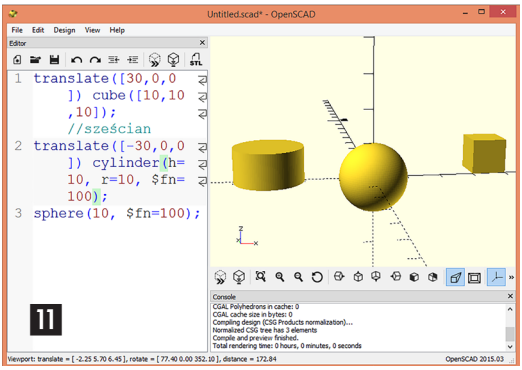












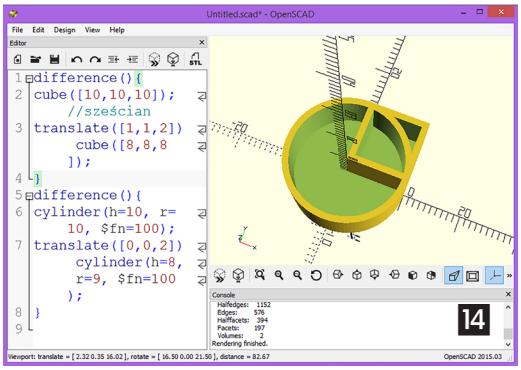
celu polecenie „translate([X,Y,Z])”, gdzie XYZ to przesunięcie w każdej z osi, a samo polecenie powinno być podane na początku komendy rysującej bryłę. Na początek należy przesunąć sześcian o 30 mm w osi X, w prawo, a następnie cylinder o 30 mm w osi X, w lewo (wartość przesunięcia musi poprzedzać znak minus). Rezultat powinien być identyczny jak na **rys. 9**, czyli bryły znajdą się po bokach, a środek będzie wolny. Zostanie w nim następnie utworzona kolejna ważna bryła, czyli kula. Do jej tworzenia należy użyć komendy: „sphere(10);”. Dzięki niej w środku układu współrzędnych powstanie kula o promieniu 10 mm, identycznie jak na **rys. 10**. Jak widać na podglądzie, zarówno kula, jak i walec mają powierzchnię złożoną z płaskich fragmentów. To skutek małej dokładności odwzorowania kształtu wektorowego, który można zwiększyć. Robi się to, dodając dodatkowy parametr – „$fn=100” (po przecinku), w walcu i kuli. Dla kuli komenda powinna więc przyjąć wartość „sphere(10, $fn=100);”, a efekt można zobaczyć na **rys. 11**. Oczywiście zwiększenie tego parametru nie jest obojętne, ma ono znaczenie podczas wydruku na drukarce 3D – bardziej dokładny kształt obiektu wydłuża czas wydruku, dlatego należy go stosować tylko tam, gdzie taka dokładność jest niezbędna.

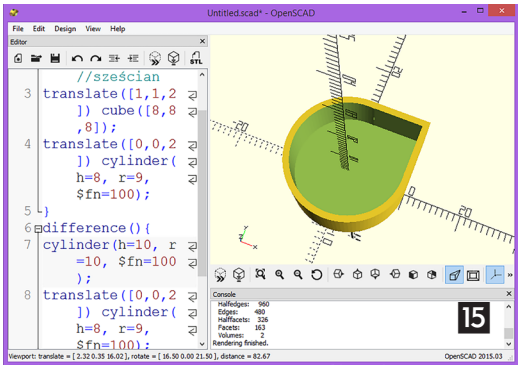
Mając cylinder, w łatwy sposób można z niego utworzyć stożek. Wystarczy „r” przemianować na „r1”, a po przecinku dodać „r2”. Wielkość parametru „r2” odpowiada za rodzaj stożka (0 to stożek ostry, inna wartość to stożek ścięty, gdzie podana wartość jest promieniem ściętego wierzchołka). Przykład komendy po zmianie: „cylinder(h=10, r1=10, r2=0, $fn=100);” a jej wynik można porównać z **rys. 12**.

Teraz nadszedł czas na **zadanie**.

Należy przy pomocy polecenia ”translate([X,Y,Z])” ustawić bryły w następującej konfiguracji: podstawka stożka powinna znaleźć się na górnej ściance sześcianu, natomiast kula powinna pojawić się na wierzchołku stożka, identycznie jak na **rys. 13**. Kod niezbędny do uzyskania podobnej konfiguracji może wyglądać następująco:

```
„translate([5,5,0]) cube([10,10,10]);
//sześcian
```






```
translate([0,0,10]) cylinder(h=10,
r1=10, r2=0, $fn=100);
translate([0,0,30])sphere(10,
$fn=100);"
```

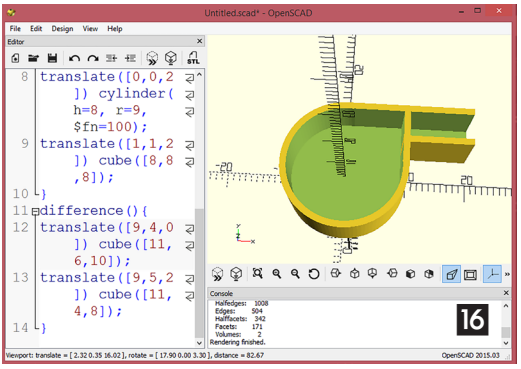
W takim przypadku najwygodniej jest przenieść wszystkie figury z powrotem do środka układu współrzędnych, przy czym sześcian dodatkowo należy przesunąć o 5 mm (połowę długości boku) w osi X i Y (znajdzie się wtedy centralnie). Następnie wystarczy stożek w osi Z podnieść o wysokość sześcianu, czyli o 10 mm, a na koniec w osi Z podnieść kulę o 30 mm (wysokość sześcianu + wysokość stożka oraz promień kuli).

Tworzymy gwizdek

Teraz warto spróbować stworzyć jakiś przedmiot praktyczny. Dość łatwo będzie zaprojektować gwizdek sędziego. Wystarczy wrócić do początku projektu – odczytać projekt1.scad. Do kolejnych kroków będzie potrzebne kolejne polecenie, usuwające zbędne części brył. Składnia tego polecenia jest następująca:

```
„difference(){
pierwsza komenda;
kolejne komendy;
....}"
```

Należy jednak pamiętać, że pierwsza komenda oznacza bryłę, z której będzie odbywało się wycinanie, natomiast kolejne komendy to bryły wycinane. Oznacza to, że z jednej bryły można wycinać inne bryły.



Na początek należy wyciąć środek walca i sześcianu, zostawiając 1 mm na ścianki gwizdka. Poprawna komenda wygląda następująco (tymczasowo bryła wycinana została podniesiona o 2 mm, aby było widać kształt wewnętrzny – ale docelowo będzie to 1 mm):

```
„difference(){
cube([10,10,10]); //sześcian
translate([1,1,2]) cube([8,8,8]); }
difference(){
cylinder(h=10, r=10, $fn=100);
translate([0,0,2]) cylinder(h=8, r=9,
$fn=100); }",
```

Wynik tego działania obserwujemy na **rys. 14**.

Jak widać, pozostały fragmenty zbędne, które można usunąć, dodając komendę użytą do wycięcia środka sześcianu do cylindra i odwrotnie.

Poprawny kod:

```
„difference(){
cube([10,10,10]); //sześcian
translate([1,1,2]) cube([8,8,8]);
translate([0,0,2]) cylinder(h=8, r=9,
$fn=100); }
difference(){
cylinder(h=10, r=10, $fn=100);
translate([0,0,2]) cylinder(h=8, r=9,
$fn=100);
translate([1,1,2]) cube([8,8,8]); }".
```

Na **rys. 15** widzimy działanie kodu. Brakuje jeszcze tzw. ustnika, który można utworzyć z prostopadłościanu przesuniętego w odpowiednie miejsce. Wystarczy do kodu na samym dole dodać kolejną komendę:

```
„translate([9,4,0]) cube([11,6,10]);".
```

Następnie trzeba z tej nowej bryły wyciąć środek, przelotowo dla osi X – wystarczy zmienić komendę do następującej postaci:

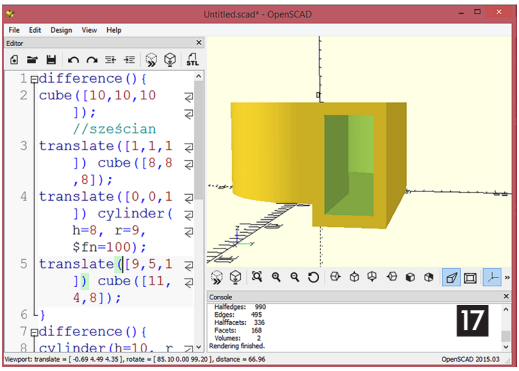
```
„difference(){
translate([9,4,0]) cube([11,6,10]);
translate([9,5,2]) cube([11,4,8]); }"
```

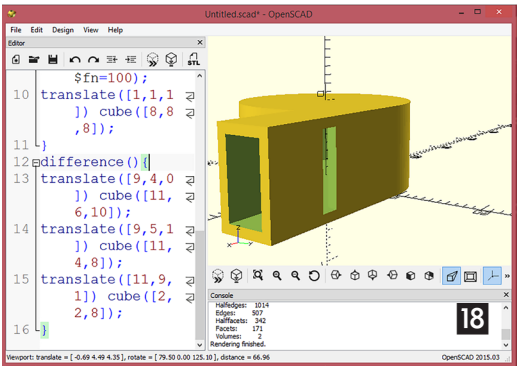
Rezultat pokazujemy na **rys. 16**. Pozostała jeszcze ścianka pomiędzy ustnikiem a komorą gwizdka. Aby ją usunąć, należy dodać bryłę wycinaną z ustnika do sześcianu, a następnie dla wszystkich "translate" w osi Z zmienić z 2 mm na wartość 1 mm.

młody m.technik
PRAKTYCZNY KURS DRUKU

Kod po zmianach powinien wyglądać następująco:

```
„difference(){
cube([10,10,10]); //sześcian
translate([1,1,1]) cube([8,8,8]);
translate([0,0,1]) cylinder(h=8, r=9,
$fn=100);
translate([9,5,1]) cube([11,4,8]); }
difference(){
cylinder(h=10, r=10, $fn=100); //walec
translate([0,0,1]) cylinder(h=8, r=9,
$fn=100);
translate([1,1,1]) cube([8,8,8]); }
difference(){
translate([9,4,0]) cube([11,6,10]); //
ustnik
translate([9,5,1]) cube([11,4,8]); }”
```





Podgląd projektu można zobaczyć na **rys. 17**. Na koniec pozostało jeszcze wyciąć szczelinę na wychodzące powietrze i gwizdek gotowy. Szczelinę trzeba wyciąć w ustniku, w osi Z, o szerokości 3 mm, z przesunięciem w osi X=11, Y=9, Z=1, np. „translate([11,9,1]) cube([2,2,8]);”.

Efekt można zobaczyć na **rys. 18**, a kod powinien wyglądać następująco:

```
„difference(){
cube([10,10,10]); //sześcian
translate([1,1,1]) cube([8,8,8]);
translate([0,0,1]) cylinder(h=8, r=9,
$fn=100);
translate([9,5,1]) cube([11,4,8]); }
difference(){
cylinder(h=10, r=10, $fn=100);
translate([0,0,1]) cylinder(h=8, r=9,
$fn=100);
translate([1,1,1]) cube([8,8,8]); }
difference(){
translate([9,4,0]) cube([11,6,10]);
translate([9,5,1]) cube([11,4,8]);
translate([11,9,1]) cube([2,2,8]); }”
```

Tak utworzony projekt można poddać **procesowi renderowania** (klawisz F6), a następnie **wyeksportować jako STL** (File => Export => Export as STL...). Uzyskany plik gwizdek.stl można następnie wczytać do programu obsługującego drukarkę VERTEX Repetier Host. ■

Jarosław Kita

W razie dodatkowych pytań, prosimy o listy na adres: jaroslaw.kita@op.pl. Autor postara się odpowiedzieć w kolejności napływających wiadomości.