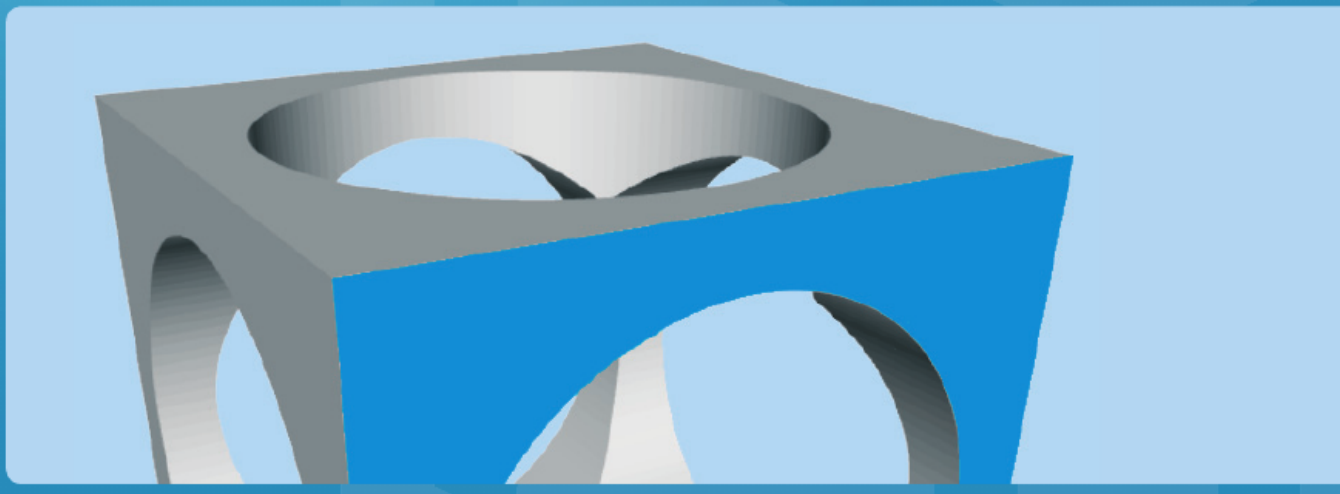
nclab
Public Computing

Solid Modeling with PLaSM

A Project-Based Approach



Revision September 12, 2013

About the Textbook

This textbook is a companion of the interactive course "Solid Modeling with PLaSM" in NCLab. With the help of many examples, review questions, and interactive exercises, this course introduces the reader to basics of 3D visualization, RGB colors, 2D and 3D shapes, geometrical transformations, and Boolean operations with geometrical objects. In its advanced part it covers the concepts of reference domains, reference mappings, and parametric curves and surfaces. This textbook along with over 100 interactive examples and exercises can be accessed through the PLaSM module in NCLab. Solution Worksheets and a Solution Manual in PDF are available to instructors.

Since the beginning, the reader is exposed to computer programming – all colors, objects, transformations and operations are defined via simple commands. The reader also learns how to utilize more advanced elements of computer programming to simplify and automate the creation of 3D designs. The combination of geometry and programming is extremely powerful and rewarding. Are you not a programmer? No worries! The language is so intuitive that there is no need to know computer programming in order to use it.

About the Authors

Dr. Pavel Solin is Professor of Applied and Computational Mathematics at the University of Nevada, Reno. He is an expert in scientific computing and the author of six monographs and many research articles in international journals.

Dr. Alberto Paoluzzi is Professor of Computer Graphics and CAD Design at the University of Rome in Italy, leader of the PLaSM project, and author of the famous monograph *Geometric Programming for Computer Aided Design, Wiley, 2003*.

Acknowledgment

We would like to thank great teachers from NCLab Partner Schools for class-testing the PLaSM module, and for providing useful feedback that is helping us to improve the textbook, interactive exercises, and the PLaSM language itself.

Graphics Design: TR-Design `http://tr-design.cz`

Table of Contents

1 Getting Started

In this section we will:

- Learn basic facts about Solid Modeling and PLaSM.
- Learn to work with the PLaSM module.
- Learn about RGB colors.

1.1 Solid Modeling and PLaSM

The word "solid" in this context means "an object", as in "square is a two-dimensional solid", "cube is a three-dimensional solid". Solid Modeling, sometimes also called 3D Modeling or 3D Design, is a collection of rules and techniques for mathematical and computer modeling of solids. It is distinguished from related areas such as computer graphics by its emphasis on *physical fidelity*. In other words, accuracy of models that are used in 3D computer games is very different from the accuracy that is required in architecture, automotive industry, and other engineering areas. Solid Modeling is the basis of computer-aided design (CAD), engineering simulations, and other disciplines.

This 3D modeling course is based on PLaSM (Programming Language of Solid Modeling), a simple and elegant scripting language with Python syntax. In fact, PLaSM is a Python library – colors, shapes, geometrical transformations and everything else is defined via simple *commands*. Entire designs are simple *Python programs*. The PLaSM module is connected to a powerful computational geometry engine on a remote server where these programs are evaluated, and resulting 3D geometries are sent back to your computer or tablet, and displayed in your web browser.

1.2 Launching PLaSM and running demo script

The PLaSM module can be launched via the CAD icon on Desktop:

Fig. 1: CAD icon on NCLab Desktop.

Alternatively, one can use the Start menu located in the bottom-left corner of the browser:

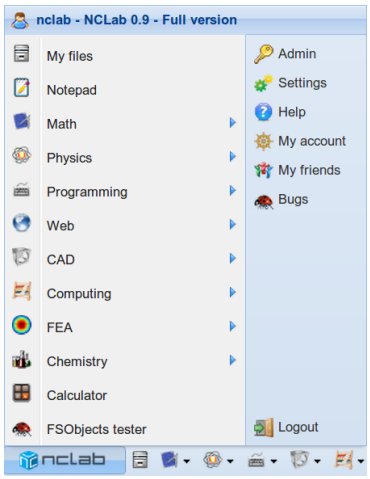


Fig. 2: Start menu.

The module opens with a demo script:

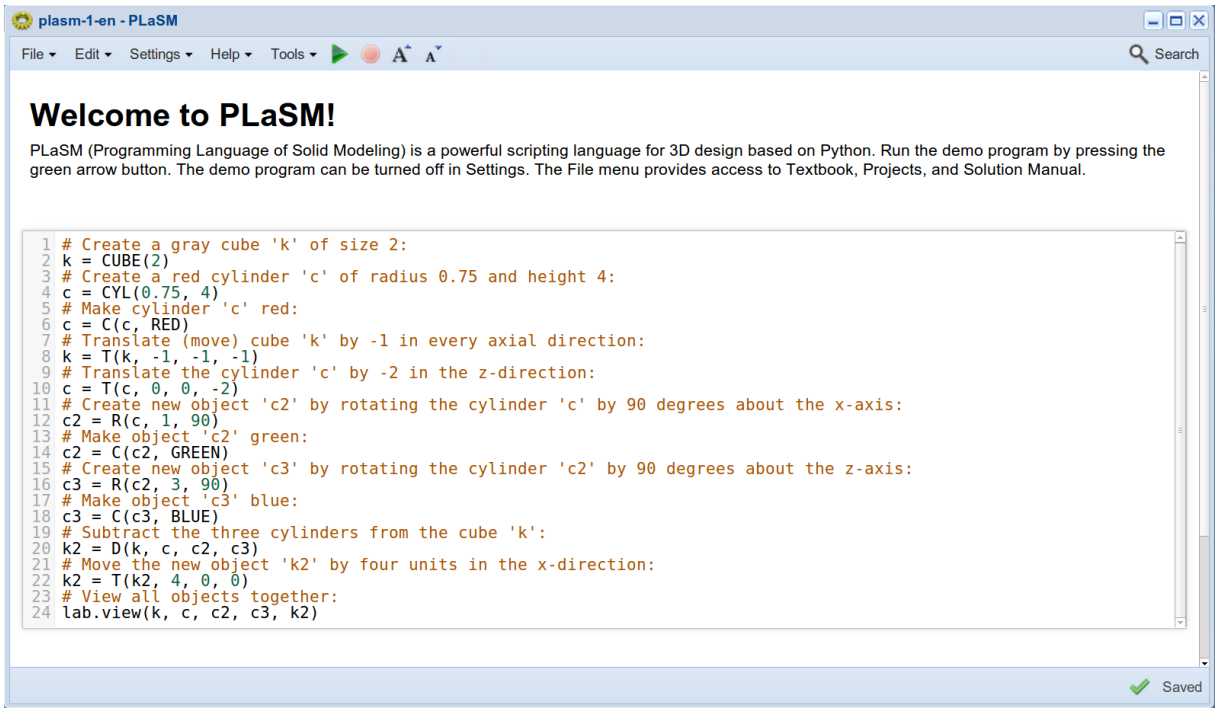


Fig. 3: PLaSM module with a demo script.

The purpose of the demo script is to illustrate to a newcomer quickly how PLaSM works. In fact, it shows a lot of functionality:

- How to create a cube.
- How to create a cylinder.
- How to translate (move) objects.
- How to rotate objects.
- How to subtract objects from each other.
- How to display objects.

Run the demo script by pressing the green arrow button, and then allow few seconds for processing on the cloud and data transfer back to your computer or tablet. The resulting geometry is shown below:

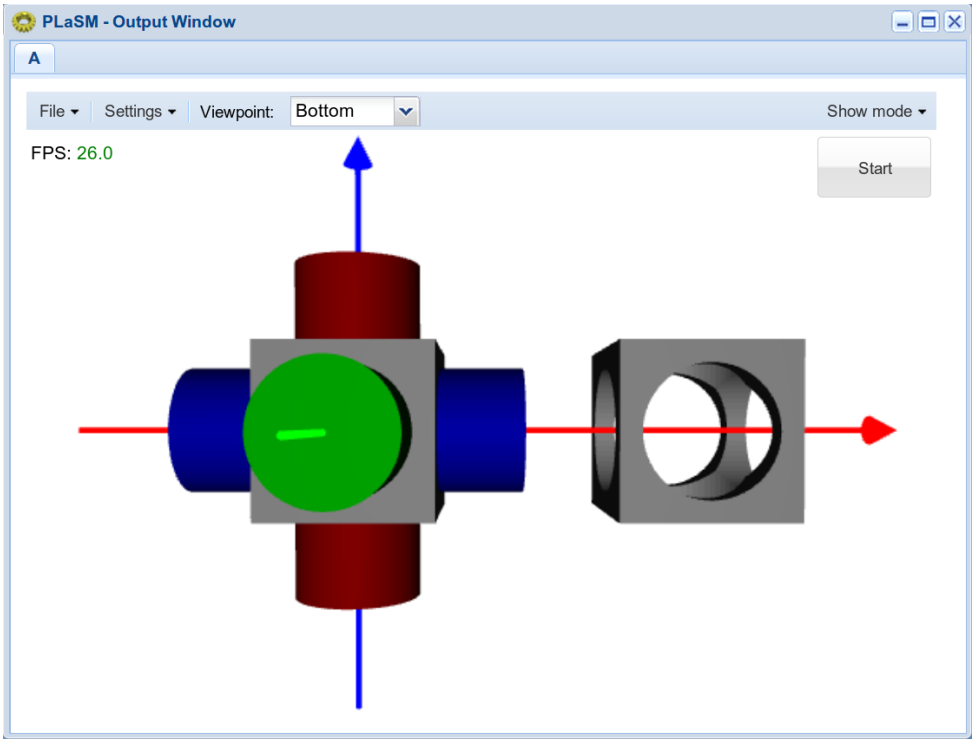


Fig. 4: Geometry created by the demo script.

Users who are familiar with Solid Modeling or have worked with another CAD system before, will probably have PLaSM figured out by now. For all others – please keep in mind that the demo script is a sneak-peek only.

1.3 3D printing

3D printing is the feature of our time, and NCLab fully supports it! Starting with Basic Plan, NCLab's users can send their designs to `support@nclab.com`. We will measure the volume and provide a quotation. The NCLab 3D printing service is non-profit and therefore up to 50% less expensive that commercial 3D printing services. See Terms of Use for details and pricing. Fig. 5 shows a 3D print of the drilled cube created by the demo script.

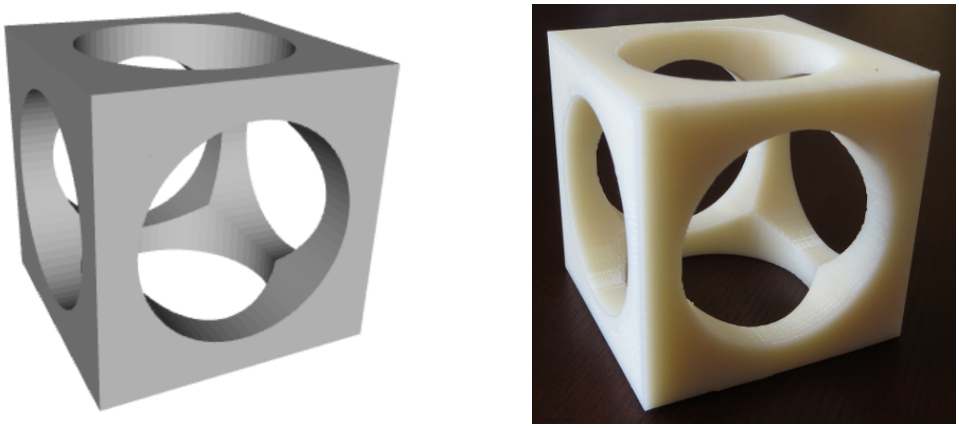


Fig. 5: 3D print of the drilled cube.

Fig. 6 shows a 3D print of a Roman temple model that we will build in Subsection 6.2.

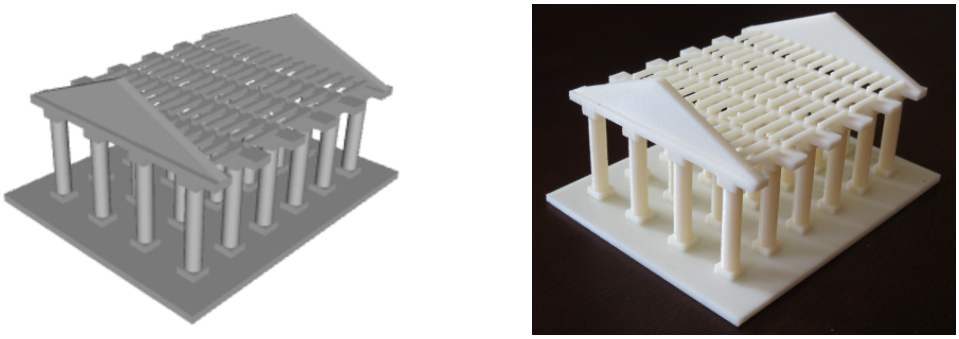


Fig. 6: 3D print of a Roman temple.

Fig. 7 shows a 3D print of a gear model that we will create in Subsection 6.4.

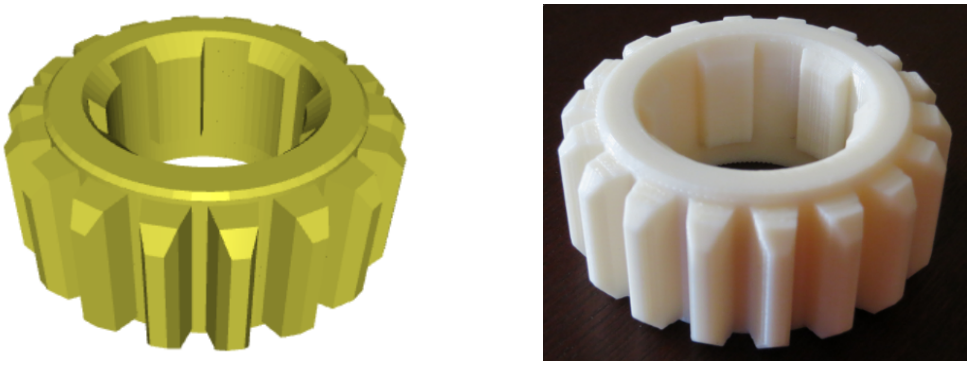


Fig. 7: 3D print of a gear model.

1.4 3D visualization

To better understand how 3D visualization works, imagine that your computer screen is a 2D plane that has a coordinate system (grid) attached to it: Horizontal axis X that lies in the plane of the screen and goes from left to right, vertical axis Y that also lies in the plane of the screen but goes from bottom to top, and a Z-axis that is perpendicular to the screen, as shown in Fig. 8.

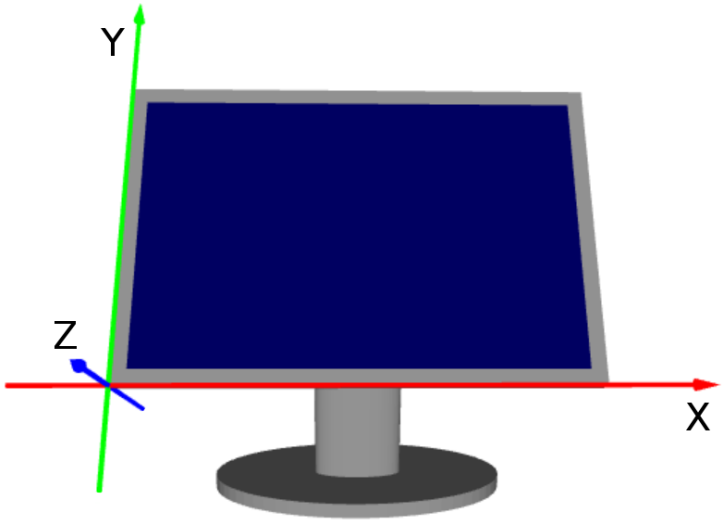


Fig. 8: Virtual grid associated with your computer screen.

The mouse can be used to rotate the object about all three axes X, Y and Z, and also to move it in all these directions. So, although the computer screen is flat, the visual experience is 3D. Before we explain how it works, run the PLaSM demo script and stop the rotation using the button in the upper right corner of the output window.

1. Turning

Hover the mouse pointer over the 3D object and press the left button. While holding it down, move the mouse to the left and to the right on the desk. The object on the screen will rotate about the Y axis. Moving the mouse up and down on the desk while holding down the left button will result in rotation about the X axis.

2. Panning

The missing rotation about the Z axis cannot be made up by combining the X and Y rotations. Therefore we use *panning*: Hover the mouse pointer over the object and press the left button again. While holding it down, start describing with the mouse small circles on the desk in the counter clockwise direction. The object on the screen will start tilting to the right (Z rotation). Similarly, circular motion in the clockwise direction will tilt the object to the left. Combined with the X and Y rotations described in point 1, now we have a complete set of all 3D rotations.

3. Moving

Holding down the middle button (or the mouse wheel) instead of the left one, and moving the mouse left to right will move the object on the screen in the X direction. Moving the mouse up and down on the desk will move the object in the Y direction. Missing the Z direction? That's why we have zooming!

4. Zooming

Zooming is done via the mouse wheel or by holding down the right mouse button and moving the mouse up and down on the desk. The object on the screen moves in the Z direction. The latter option is smoother. Hence, with zooming we have a complete set of 3D motions on the computer screen.

The mouse controls only adjust the view of the model in the X, Y and Z coordinates associated with your screen. They do not make any changes to the model itself.

1.5 RGB colors

According to the *additive color model* which is based on the human perception of colors, every color can be obtained by mixing the shades of Red, Green and Blue (R, G and B). These are called *additive primary colors* or just *primary colors*. The resulting color depends on the proportions of the primary colors in the mix. It is customary to define these proportions by an integer between 0 and 255 for each primary color. So, the resulting color is a triplet of real numbers between 0 and 255:

Color	R	G	B
Red	255	0	0
Green	0	255	0
Blue	0	0	255

These colors are shown in the following Fig. 9.

Fig. 9: Pure red [255, 0, 0], pure green [0, 255, 0], pure blue [0, 0, 255].

When the proportions of all three primary colors are the same, the result is a shade of grey. With [0, 0, 0] one obtains black, with [255, 255, 255] white:

Fig. 10: Black [0, 0, 0], dark grey [128, 128, 128], light grey [220, 220, 220].

Guessing RGB codes is not easy. If you need to find the numbers representing your favorite color, the best way is to google for "rgb color palette". You will find many pages that translate colors into RGB codes. Fig. 11 shows three "easy" colors cyan, pink and yellow along with their RGB codes.

Fig. 11: Cyan [0, 255, 255], pink [255, 0, 255], yellow color [255, 255, 0].

In Subsection 2.2 we will learn how colors are assigned to objects in PLaSM. In Subsection 3.1 we will see how various objects in one scene can be colored differently.

2 Library of Simple Shapes

In this section we will learn to

- Create a variety of 2D and 3D shapes.
- Assign colors to objects.
- Create convex hulls and extrude 2D objects to 3D.

This section serves mainly as a *database of shapes* and it is intended for reference rather than for systematic study. You may want to browse through it quickly to get an overview what shapes are available, but our goal is to move to design projects quickly. Note in Subsection 2.2 how colors are assigned to objects. Your first project is awaiting you in Section 3 and it will only require cubes and bricks.

2.1 Cube

The command `CUBE(a)` creates a cube of dimensions $a \times a \times a$. Every newly created object comes in a default steel color. To visualize objects, we use the `VIEW` command, that can be abbreviated by `V`. Type the two lines below into the PLaSM input cell and press the green arrow button:

```
c = CUBE(1)
VIEW(c)
```

After a moment, the following image should appear in your web browser:

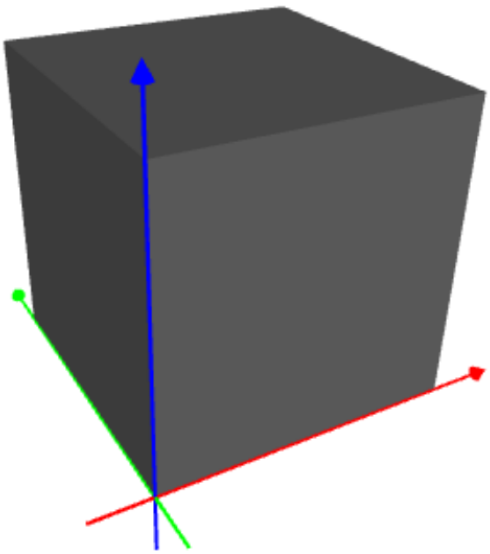


Fig. 12: Steel cube of dimensions $a \times a \times a$ with $a = 1$.

Note that the cube is located in the first quadrant, with its edges aligned with the coordinate axes x (red), y (green) and z (blue). One of its vertices lies at the origin $(0, 0, 0)$. Every cube created via the `CUBE` command will be positioned like this.

2.2 Two ways to color objects

The command `COLOR`, possibly abbreviated as `C`, assigns a given RGB color to a 2D or 3D object. Its use is best illustrated on the previous example, where we change the color of the cube from steel to brass:

```
c = CUBE(1)
c = C(c, BRASS)
VIEW(c)
```

Note that PLaSM keywords are written in *capital letters*. This is to avoid conflicts with user-defined variables (names of objects, custom colors, etc.). For newcomers to programming – in the above program, `c` is a user-defined variable (intentionally chosen to be lowercase), and `CUBE`, `C`, `BRASS` and `VIEW` are PLaSM keywords. After executing the script, you will see the following:

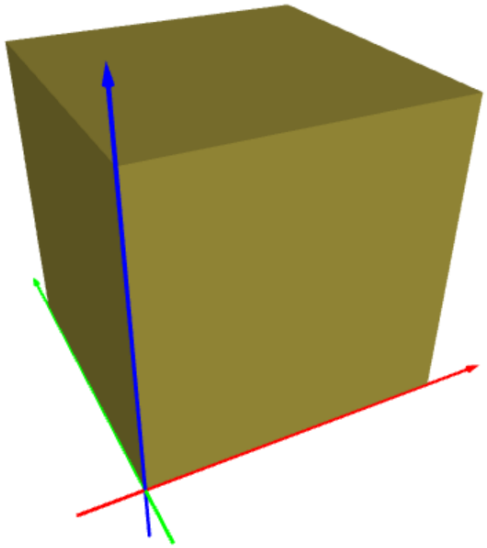


Fig. 13: Same cube as before, but now in brass color.

Note that the axes in the grid are color-coded using the word "RGB" for convenience: x = R, y = G, z = B.

There is another possible way to display the cube `c` in brass color:

```
c = CUBE(1)
VIEW(c, BRASS)
```

This program is shorter but it does a different thing: The cube keeps its default steel color while the VIEW command just shows how it *would look like* in brass color. This is a great shortcut for experiments, but the former approach is preferable in real designs.

The keyword `BRASS` is just a predefined triplet of numbers `[255, 250, 83]`. PLaSM offers the following predefined colors:

```
GREY = [128, 128, 128]
GREEN = [0, 255, 0]
BLACK = [0, 0, 0]
BLUE = [0, 0, 255]
BROWN = [139, 69, 19]
CYAN = [0, 255, 255]
```

```
MAGENTA = [255, 0, 255]
ORANGE = [255, 153, 0]
PURPLE = [128, 0, 128]
WHITE = [255, 255, 255]
RED = [255, 0, 0]
YELLOW = [255, 255, 0]
```

and metallic colors:

```
STEEL = [255, 255, 255]
BRASS = [181, 166, 66]
COPPER = [184, 115, 51]
BRONZE = [140, 120, 83]
SILVER = [230, 232, 250]
GOLD = [226, 178, 39]
```

2.3 Planar square

PLaSM makes it possible to work in a two-dimensional setting of the axes x and y (in addition to 3D). Here, the z axis is not present. The first 2D command that we will explore is `SQUARE(a)` which renders a square of edge length a. Its usage is illustrated by the following script:

```
s = SQUARE(3)
s = C(s, BRASS)
VIEW(s)
```

The square is shown in Fig. 14. Again note where it is positioned - all newly created squares are positioned like this.