# TNM067 — Scientific Visualization
# 1. Introduction to VTK

## 1 Introduction

Methods and algorithms that we are getting familiar with in this lab are the following:

- Construct simple VTK applications
- Use the visualization pipeline
- Read data from file
- Generate data from the wave function of a hydrogen atom using implicit functions.

VTK references:

- `http://vtk.org/doc/release/5.8/html/`
- `http://www.vtk.org/`
- The Visualisation Toolkit, 4th Edition
- The VTK User's Guide (optional)

Important vocabulary

- Visualization pipeline
- Implicit functions

A thing to keep in mind for this and the following labs is that VTK is strongly object oriented, so sometimes you find the solution through inherited structures. Also watch the data types for arguments as well as return types and make sure you use the correct types, some problems might be solved through type casting.

### 1.1 Introduction to Program Arguments

Since you are going to investigate different algorithms and methods it is suitable to be able to control the program with arguments from the command prompt. Arguments to the program are gathered in an array (`argv`) along with a number (`argc`) that indicate how many the arguments are in this array. Here is an example of how you can call this program from the command prompt:

```
prompt% ./prog -arg1 -arg2 data -arg3
```

And the content in these variables will look like this:

```
argc = 5
argv[0] = "./prog"
argv[1] = "-arg1"
argv[2] = "-arg2"
argv[3] = "data"
argv[4] = "-arg3"
```

To get hold of these variables you are *encouraged* to use `getopt`, which is a Unix command, use the Unix manual to find out more about it. [Note: There is no real equivalent to `getopt` in MS Windows although various solutions exist.] This example shows you how to use it:

```
extern char * optarg;
extern int optind;
int c;
int error = 0;
while ((c = getopt(argc, argv, "abc:")) >= 0) {
  switch(c) {
  case 'a':
    aopt = 1;
    break;
  case 'b':
    bopt = 1;
    break;
  case 'c':
    copt = 1;
    strcpy(cdata,optarg);
    break;
  case ':':
    error = 1;
    printf("Missing value for argument - %s\n", argv[optind]);
    break;
  case '?':
    error = 1;
    printf("Invalid argument - %s\n", argv[optind]);
    // Could print usage list here...
    break;
  default:
    error = 1;
    printf("Invalid argument state: %d\n", c);
    break;
  }
}
if (error)
  exit(1);
// Now handle meaning of arguments...
```

The third argument to `getopt` is a string that identifies the different arguments that are accepted by the program. A colon after a character implies that the argument requires a following option. Note that the function is case-sensitive, `a` and `A` are two different arguments. The code above can be used with this command line for the program:

```
prompt% ./prog -b -c kalle -a
```

# 2 Getting started

VTK SDK is installed in the system at this location: `/usr/include/vtk-5.8`. If you decide to write your own Makefile or to compile some project manually, make sure you include the pathway to the header library as a compilation argument: `-I/usr/include/vtk-5.8`.

The framework for the later part of this lab is accessible from the course homepage. These files are:

**Makefile** will make your life easier when it comes to compiling VTK. Make sure you use it.

**Cone.cpp** is an example from the lecture. Read the file and make sure to understand what is done to produce the rendered image. Note that the method `New()` is used instead of the operator `new`. Instead of the operator `delete` the method `Delete()` is used. This method is counting down the number of references to an object, not deleting it until the number reaches zero.

**ImpFunc.cc** This contains the code outline for the implicit function definition.

## 2.1 Compile and run

To compile:

```
prompt# make Cone
```

An executable file is produced. To run this program:

```
prompt# ./Cone
```

# 3 Note on VTK

VTK uses *forward declaration* (declaring a type for the purpose of providing an interface or protocol for the programmer without writing what the type does). This can cause compiler errors if one tries to cast a type that is not *fully declared*. To prevent this from occurring, one must include the header file which contains the full declaration of the type.

In short, always remember to include the header file of the type that you will use in your program, i.e., if you use `vtkLight`, you must add `#include <vtkLight.h>` to your program.

# 4 Interaction

A static image is produced in the example above. To better make use of the visualization you will add interactivity to the scene.

`vtkRenderWindowInteractor` provides means for interaction in VTK. Read about it in the manual pages. What functionality besides window interaction does it provide?

Create an instance of the class and associate it with `vtkRenderWindow` by using `SetRenderWindow(...)`. When you have defined the whole pipeline, you start the interaction by calling

`Start()`. Observe that the visualisation pipeline is executed backwards; it is the `vtkRenderWindowInteractor` that sends an update call through the pipeline. When you have implemented an interactor you can remove the lines of code that are waiting for a keystroke.

**Task 1 — Add interaction:**
Add interaction to `Cone.cpp`. Read the manual pages about `vtkRenderWindowInteractor` and `vtkInteractorStyle`. Which is the default interactor style? Change the interactor style such that it behaves like a trackball on startup. What other interactor styles exist in VTK? Try out the different mouse and keystrokes that `vtkRenderWindowInteractor` supports. Get yourself familiar with the structure of the manual pages.

## 5 Object properties

`vtkActor` inherits methods for rotation, translation and scaling, as well as for modifying color, material features, etc, from `vtkProp3D`. Change the material features of the cone. What other subclasses does `vtkProp3D` have and what do they do?

## 6 Interact with the object and update the pipeline

Use the wxGUI.cpp from the lab directory for this part of the lab. Compile the program like this:

prompt% make wxGUI

The program wxGUI.cpp has a graphical user interface (GUI), built in wxWidgets, that modifies properties in the pipeline. When a feature is chamged the pipeline is executed again. Read the code and the manual pages about vtkConeSource. See to it that you understand how the interaction works.

**Task 2 — GUI interaction:**
Modify the code in wxGUI.cpp such that you can interactively change the color of the cone.

## 7 Read in a file in the standard format

VTK has support for a number of different file formats. Look for the classes that end with "Reader" and "Importer" in the Class Hierarchy list in the manual pages. You will load a 3D model from a stereo-lithography file (`42400-IDGH.stl`), which can be specified with the method `SetFileName(...)`.

**Task 3:**
Copy `Cone.cpp` to a new file `Read.cpp`. Replace the cone with the data in the file `42400-IDGH.stl`. Make changes in the `Makefile` to compile `Read.cpp` using the command:

```
prompt% make Read
```

# 8 Wave Function of a Hydrogen Atom

In this exercise you will do some physics and use the wave function of a hydrogen atom (single electron) in the state $2p_0$. From the wave function it is possible to calculate the probability of finding an electron at each position in space, making most sense while not too far from the atom's nucleus. In this lab you will just generate the data, in the next lab you will create a visualization using isosurfaces. An implicit function will be used to sample the electron pobability density function.

## 8.1 The Wave Function

The wave function consists of two parts: the radial function $R_{nl}(r)$ and the spherical harmonics $Y_{lm}(\theta, \phi)$ (providing the angular part). The subscripts $n$, $l$, $m$ are called *the three principal quantum numbers* and in this case you will use the wave function for $n = 2$, $l = 1$ and $m = 0$.

Using spherical coordinates:

$$R_{2,1}(r) = \frac{1}{\sqrt{3}} \left(\frac{Z}{2a_0}\right)^{3/2} \left(\frac{Zr}{a_0}\right) e^{-\frac{Zr}{2a_0}} \tag{1}$$

$$Y_{1,0}(\theta, \phi) = \sqrt{\frac{3}{4\pi}} \cos\theta \tag{2}$$

giving the complete wave function:

$$\Psi_{2,1,0}(r, \theta, \phi) = \frac{1}{4\sqrt{2\pi}} \left(\frac{Z}{a_0}\right)^{3/2} \left(\frac{Zr}{a_0}\right) e^{-\frac{Zr}{2a_0}} \cos\theta \tag{3}$$

from which the electron probability density function is calculated as

$$P_{2,1,0} = |\Psi_{2,1,0}(r, \theta, \phi)|^2 \tag{4}$$

In your code, do you have to calculate the absolute value of $\Psi_{2,1,0}(r, \theta, \phi)$? The constant $a_0$ is the *Bohr radius* (0.53Å) and $Z$ is the charge of an electron. Because this case regards a single electron hydrogen, you should set:

$$Z = 1, \quad a_0 = 1 \tag{5}$$

If you feel the urge to read more about the wave function and the probability density function you can do so at http://academic.reed.edu/chemistry/roco/Density/cloud.html or read the book *Physics of Atoms and Molecules* by B.H. Brandsen and C.J. Joachain (which was the source for the equations used here). You can also try the applet at http://physics.ius.edu/˜kyle/physlets/quantum/hydrogen.html to get an indication of what your result should look like.

There is a set of equations which enables conversion between the Cartesian coordinate system and the spherical coordinate system. The angles $\phi$ and $\theta$ are easily mixed up because the notations are sometimes different in maths and physics, the notation here is that of physics. You can read more about spherical coordinates at http://mathworld.wolfram.com/SphericalCoordinates.html. To go from the spherical coordinate system to the Cartesian coordinate system, use:

$$x = r \sin\theta \cos\phi \tag{6}$$

$$y = r \sin\theta \sin\phi \tag{7}$$

$$z = r \cos\theta \tag{8}$$

To go from the Cartesian coordinate system to the spherical coordinate system, use:

$$r = \sqrt{x^2 + y^2 + z^2} \quad (9)$$
$$\theta = \cos^{-1}(z/r) \quad (10)$$
$$\phi = \tan^{-1}(y/x) \quad (11)$$

## 8.2 Implicit Function

To create an implicit function one has to define a subclass to `vtkImplicitFunction`. The file `ImpFunc.cc` contains a code stubb for such a class but you need to implement the function EvaluateFunction(double x[3]) (using the equations for the wave function). The following code shows you the basics of how to use the `ImpFunc`.

```
...
ImpFunc * function = ImpFunc::New();
vtkSampleFunction * sample = vtkSampleFunction::New();
sample->SetImplicitFunction(function);
sample->SetModelBounds(...);
sample->SetSampleDimensions(...);
sample->ComputeNormalsOff();
...
```

Either modify one of your previous files or make a new one to include the code for writing the sampled implicit function to a file.

**Task 4 — Implicit Function:**
Implement the wave function and then let it be sampled in a Cartesian coordinate system with the resolution $32 \times 32 \times 32$. The origin should be in the centre of the volume with range $[-6, 6]$ in each dimension. Write the data to a file called "hydrogen.vtk". You have to figure out how to use the probability density function to generate the data. You can use the `vtkStructuredPointsWriter` to write the data to a file. Look at the file that you created and see how it is structured.

To see if you have implemented the implicit function correctly run the HydrogenTest program. It will read the "hydrogen.vtk" file you created and print out Success if it is correct or Failure if it is not.

The "hydrogen.vtk" file will be used in the next lab.