Creating computer graphics and animations based on parametric equations of lines and curves – proposals for mathematics education at upper secondary level

Andreas Filler
filler@math.hu-berlin.de
Institut für Mathematik
Humboldt-Universität zu Berlin
10099 Berlin
Germany

Abstract

Creating computer visualizations, especially animations, can help students to understand geometric objects, which are described by parametric equations, as point sets and to discover functional relationships and dynamic aspects. Because creating computer animations is very attractive for students it can help to motivate them to figure out features of parametric descriptions.

This paper makes proposals for creating graphics and animations on lines, circles, spirals, trajectory parabolas, cycloids and other curves by describing these curves with parametric equations and shows some examples created by students at upper secondary level.

Computer animations based on parametric equations of lines and curves can be created using computer algebra systems or photorealistic 3d graphics software (like POV-Ray). Examples using both kinds of software will be shown and described.

About this article: Watching embedded animations and using files

This paper contains some animations which can be played back directly inside the article. To watch the animations, the article has to be opened with Adobe Reader or Acrobat Version 9 or later. (Alternative PDF-Software like Sumatra PDF or FoxIt Reader isn't capable to play animations.)

All graphics and animations in this article were created using the graphics software POV-Ray or the Computer Algebra System (CAS) MuPAD. POV-Ray is Freeware and can be downloaded from http://www.povray.org. MuPAD is available as part of the symbolic math toolbox add-on for MatLab. The source files which were used to create the graphics and animations can be downloaded by links which are provided in the the article, see also [References]. There are three kinds of files:

- Files with the extension `.mn` can be used with the CAS MuPAD.
- POV-Ray scripts (scene description files) have the extension `.pov` and can be opened, changed and rendered with the script driven graphics software POV-Ray.

- Files with the extension `.ini` are initialization files for creating animations using POV-Ray. To do this, an .ini- and a corresponding .pov-file have to be created (or downloaded) in the same folder. By rendering an .ini-file, a sequence of images will be generated which can be converted into a video with additional software like pjbmp2avi.

A short introduction into creating graphics using POV-Ray is available on the webpage of the author; comprehensive tutorials contains the webpage of Friedrich A. Lohmueller.

1 Introduction

Introductions to parametric equations in analytic geometry lessons in schools are usually quickly followed by assignments on transformation of parameters in coordinate form and vice versa as well as the study of positional relationships and calculations of intersecting points. Two important and related aspects of parametric equations are often disregarded:

- Students acquire only a rudimentary perception of geometrical objects as point sets.
- Students mostly fail to recognize functional relationships between parameter values and associated points. Recognition of this type of relationships requires to understand the dependence of the locations of points in space from parameters.

To prevent the emergence of limited concepts of parametric equations and to embrace the point set notion and focus on functional relationship, there are two approaches:

- Students construct the points belonging to several parameter values in an equation of the kind $P = P_0 + t \cdot \vec{a}$ and recognize that these points lie on a straight line. Parametric equations of straight lines can be introduced based on this perception. Even the description of various curves is possible in this way. Furthermore, reverse considerations and comparisons of various parameterizations of the same objects can be done.
- The dynamic view of straight lines and curves as paths can be highlighted whereby students link a concrete meaning to parameters. The interpretation of the parameter as time establishes relationships with the description of motion in physics and makes it possible to create computer animations (videos) using suitable software (as described later).

2 Using graphics software and/or Computer Algebra Systems for discovering features of parametric equations

2.1 Straight lines as point sets

To assemble straight lines or curves as point sets using parametric equations and to be able to create parameter-dependent animations, computer algebra systems (CAS, e. g. Mathematica, Maple and MuPAD) as well as 3D-graphics software, POV-Ray, among others, can be used.[1]

[1] For more about the use of 3D computer graphics software in mathematics education please see [Filler/Rieper, 2004], [Filler, 2007], [Krumpe, 2005] and [Wells et al., 1993].

Introduction of parametric equations by considering individual points. To introduce parametric equations of straight lines students can be given the following type of assignment:

Given the point $P(0.5; 1; 1.5)$ and the vector $\vec{a} = \begin{pmatrix} -2.5 \\ 1 \\ -1.5 \end{pmatrix}$.

- Represent P as well as $\vec{a}$ (as an arrow, starting from P).
- Represent the points $P + 0.5\,\vec{a}$, $P + \vec{a}$, $P + 1.5\,\vec{a}$, $P + 2\,\vec{a}$ as well as $P - 0.5\,\vec{a}$, ..., $P - 2\,\vec{a}$.
- View the representation from various directions.

Figure 1 a) shows a solution to this assignment using POV-Ray (with extensions anageoL.inc for the simple representation of objects of analytic geometry like arrows, lines, planes, coordinate systems and others). The following commands are entered:

```
#declare a = <-2.5,1,-1.5>;
#declare P = <0.5,1,1.5>;
pluspunkt(P, schwarz)
vektoranpunkt(P, a, silbergrau)
```

```
punkt(P-2*a, blau_matt)
punkt(P-1,5*a, blau_matt)
   ...
punkt(P+1,5*a, blau_matt)
punkt(P+2*a, blau_matt)
```

The entire code is contained in the file pareq-line1.pov which can, in conjunction with the extension file anageoL.inc, be used to create and modify an illustration as shown in Figure 1 a).

After representing larger numbers of points by reducing parameter distances it is getting obvious that all points of the straight line going through P whose direction is given by $\vec{a}$ can be represented with $t \in R$ in the form $P + t \cdot \vec{a}$.

Using loops to represent large numbers of points. By generating very large numbers of points, the results can no longer be visibly differentiated from straight lines or pathways. Thereby students can get a "plastic impression" of the point set character of geometric objects. The graphics represented in Figures 1 b) and c) can be created in POV-Ray using loops:

```
#declare i=-200;
  #while (i <= 200)
    punkt(P+i*a/100  blau_matt)
  #declare i=i+1;
  #end
```

The complete code is contained in the file pareq-line2-while.pov. Provided the condition $i \leq 200$ is met, this command creates a small sphere with the centre point $P + (i/100) \cdot \vec{a}$ and the value of the loop variables i is raised by 1. Any number of "points" can be represented by changing the values 200 and 100.

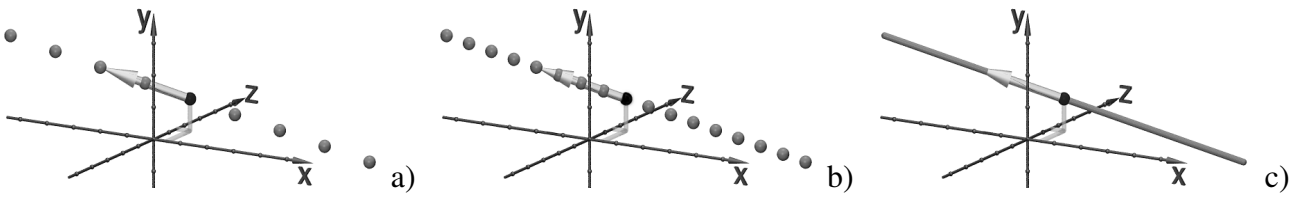


Figure 1: Points of a straight line, graphics generated with POV-Ray

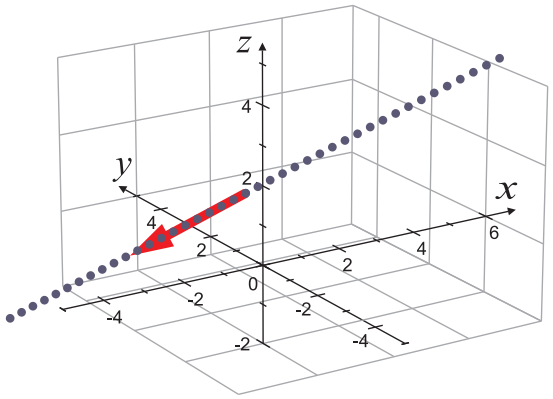


Figure 2: Points of a straight line, generated using a procedure in the CAS MuPAD

Creating lines as point sets using a Computer Algebra System (CAS). It is also possible to use a CAS instead of POV-Ray to represent straight lines (which are described by parametric equations) as point sets. Figure 2 was created using the CAS MuPAD (File: pareq-line-mupad.mn):

```
Puenktchen := proc(i)
begin
  plot::Point3d(P+i/10*a);
end_proc:
plot(Puenktchen(i) $ i = -20..20)
```

By changing the values 10 and 20 arbitrary numbers of points can be generated.

2.2 Time as parameter – generating simple videos

To create animations coordinates can be expressed depending on a time parameter (in POV-Ray `clock`). For example a video with straight-line uniform motion of a sphere can be generated by

```
#declare a = <-2.5,1,-1.5>;
#declare P = <0.5,1,1.5>;
sphere{ P+2*clock*a  0.1
        texture{pigment{color Blue}}}
```

- POV-Ray-files for the animation: animation1.pov and animation1.ini (also anageoL.inc is needed)
- MuPAD-File for a similar animation: animation1.mn

Figure 3: Simple animation of a spere on a line

The motion track becomes clearer when the track of the moving object is displayed simultaneously as pathway between the starting point and the respective position reached. This is possible using the already described illustration of straight lines or pathways as point sets.

- POV-Ray-files for creating the animation as shown in Figure 4:
  anim-lines.pov and anim-lines.ini

Figure 4: Uniform and accelerated motions (Video)

Parametric equations in animations acquire an aspect that does not affect the geometric shape of these objects: the velocity of motion. For example the two parametric equations

$$(1) \qquad P(t) = P_0 + t \cdot \vec{a} \quad (t \in \mathbb{R}_+) \quad \text{and}$$
$$(2) \qquad P(t) = P_0 + t^2 \cdot \vec{a} \quad (t \in \mathbb{R}_+)$$

describe the same half-line. If these parametric equations are used to generate animations, then (1) yields a uniform and (2) a constantly accelerated motion. This can be recognized in Figure 4 through the distances of the points; the same amount of time elapses between two adjacent points.

2.3 Vector parametric equations – the trajectory parabola

If vector descriptions come to the fore in the lessons, one can explore the above-mentioned aspect of velocities of motions. Animations allow to establish links to the physics lessons, work out functional aspects by considering various functions $f(t)$ which replace the time parameter as well as create simple simulations. An example is the oblique projection. This can be interpreted as motion assembled from uniform motion and constantly accelerated motion:

$$\vec{x} = \vec{x}_0 + \vec{v} \cdot t + \tfrac{1}{2}\vec{g} \cdot t^2.$$

This equation describes the trajectory parabola. A corresponding animation (see Figure 5) can be generated in POV-Ray by means of the following commands:

```
#declare x0 = <-2.5,0,0>;
#declare v0 = <5,5,0>;
#declare g = <0,-10,0>;
sphere { x0 + v0*clock + g/2*clock*clock  0.25}
```

- POV-Ray-files for creating the animation as shown in Figure 5:
  trajparabola.pov and trajparabola.ini
- MuPAD-File for a similar animation:
  trajparabola.mn

Figure 5: Trajectory parabola (Video)

3 Parametric equations of circles and related curves

Representations of geometric objects by parametric equations are not limited to linear objects. Circles are well suited as starting point for describing interesting curves through parametric equations and for generating animations based on it.

3.1 Parametric equations of circles

Upper secondary level students are usually already familiar with the sine and cosine function on the unit circle, with the designations used in Figure 6 as

$$\sin\alpha = y_\alpha; \qquad \cos\alpha = x_\alpha.$$

A generalization on circles in centre location with any radius r is easily possible from which the parametric equation

$$x(\alpha) = r \cdot \cos\alpha; \qquad y(\alpha) = r \cdot \sin\alpha \qquad \text{with } \alpha \in [0; 2\pi)$$

can be derived for a circle of the plane whose centre point lies in the origin. If different values have to be animated, it makes sense to standardize the interval of the parameter (time):

$$x(t) = r{\cdot}\cos(2\pi{\cdot}t); \quad y(t) = r{\cdot}\sin(2\pi{\cdot}t) \quad \text{with } t \in [0; 1).$$

Parametric equations of circles which lie in space on coordinate planes or planes parallel to them arise by representing one of three coordinates as constants, for example, $y(t) = h$. With these considerations, students can create animations of circular motions. In POV-Ray the command

```
sphere{<10*cos(2*pi*clock), 0, 10*sin(2*pi*clock)>  1 }
```

creates the animation of a sphere on a circular trajectory (see Figure 7). The motion track can be depicted by generating a multitude of small spheres (as already described for straight lines).

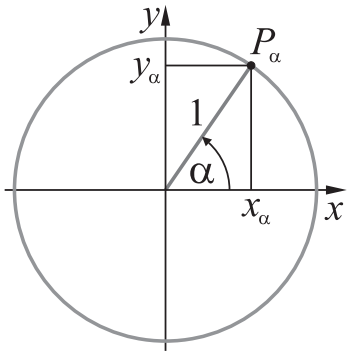


Figure 6: Sine and cosine on the unit circle

Figure 7: Motion of a small sphere on a circle (Video)

POV-Ray-files for creating the animation as shown in Figure 7: anim-circle.pov and anim-circle.ini.

3.2 Camera animations

If instead of a geometric object the position of a ”camera” is described as time-dependent, we get an animation where the view on a scene changes. For example, with the POV-Ray-Command

```
camera{ location <r*cos(2*pi*clock), 4, r*sin(2*pi*clock)>
        angle 12
        look_at <0,0,0> }
```

a camera flight on a circular path where the camera remains directed at the origin of coordinates can be simulated, see the video in Figure 8. Regarding the necessary mathematical considerations, it does not matter whether students create the animation of an object moving on a trajectory or a camera animation where the view of an entire scene changes. From experience it is known that the latter is more interesting for many students. However, to make motion curves visible, it is recommended to not only create camera animations but also to animate visible objects.

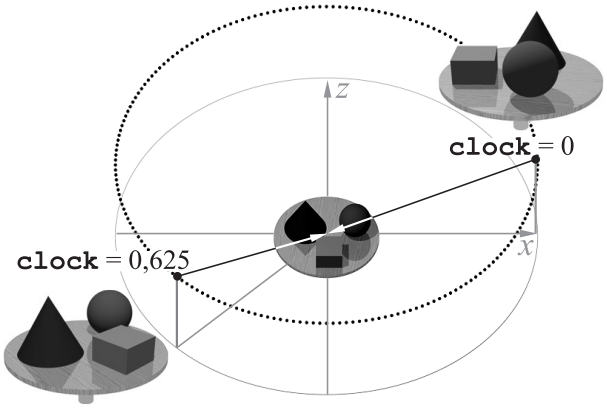


Figure 8: Camera animation on a circle (Video)

POV-Ray-files for the animation (Figure 8): camera-anim-circle.pov and camera-anim-circle.ini. A similar animation can be generated in the CAS MuPAD by the file camera-anim-circle-MuPAD.mn.

3.3 Variations of circles: spirals and helices

Interesting curves can be derivated from parametric equations of circles by functional considerations. In connection with camera animations students often ask the following questions:

1. How can the camera circle around an object and simultaneously approach it?
2. In a circular motion, how can the camera simultaneously change its height so that objects can be viewed from different heights?

Both questions can also be formulated so that they pertain to the course of curves. In order to realize the first property students can work out that the constant r, which was used for the radius of the circle,

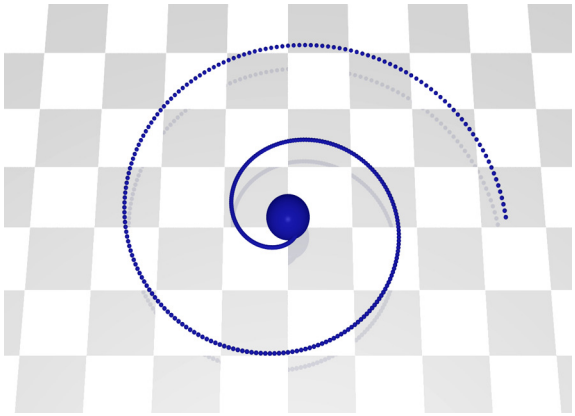


Figure 9: Archimedes' spiral

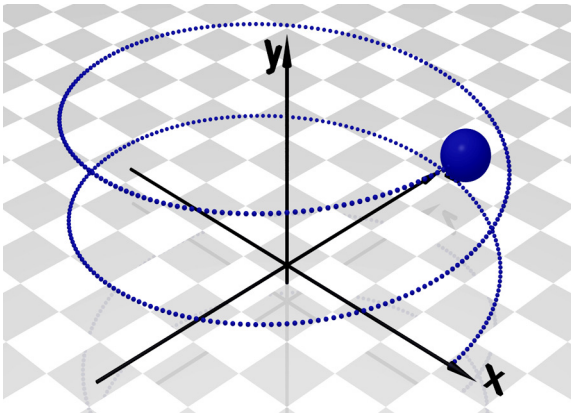


Figure 10: Helix

is replaced by a function $r(t)$ of the temporally changing parameter t, e. g. by $r{\cdot}(1{-}t)$, if the distance to the centre point during an animation should decrease from r to 0 (for $t \in [0;1]$). This notion yields the *parametric description of an Archimedes' spiral*:

$$\begin{aligned} x(t) &= r \cdot (1{-}t) \cdot \cos(2\pi{\cdot}t) \\ y(t) &= h \qquad\qquad\qquad\qquad t \in [0;1] \\ z(t) &= r \cdot (1{-}t) \cdot \sin(2\pi{\cdot}t)\ . \end{aligned}$$

On this basis, objects or the camera can move along an Archimedes' spiral. To run two revolutions (as shown in Figure 9) $(2\pi{\cdot}t)$ was replaced with $(4\pi{\cdot}t)$.

In the discussion of question 2 above, students may easily recognize that for the time-dependent alteration of the "height" the previously constantly held third coordinate has to be replaced with a function of the parameter. If a linear function is used (e. g. in the easiest case $y = t$), then the circle equation yields the equation of a *helix* (see Figure 10):

$$\begin{aligned} x(t) &= r \cdot \cos(4\pi{\cdot}t) \\ y(t) &= t \qquad\qquad\qquad t \in [0;1] \\ z(t) &= r \cdot \sin(4\pi{\cdot}t)\ . \end{aligned}$$

By combining the two considerations which led from the circle to the spiral or to the helix we get a *conical spiral* with a parametric equation of the form

$$\begin{aligned} x(t) &= r \cdot (1{-}t) \cdot \cos(4\pi{\cdot}t) \\ y(t) &= t \qquad\qquad\qquad\qquad t \in [0;1] \\ z(t) &= r \cdot (1{-}t) \cdot \sin(4\pi{\cdot}t)\ . \end{aligned}$$

In Figure 11 a video of an animated conical spiral is shown (Source Files: anim-conicspiral.pov and anim-conicspiral.ini). A similar animation can be generated in MuPAD using the file anim-conicspiral.mn. Moving a camera on a conical spiral leads to interesting camera animations (remember the questions 1 and 2 at the beginning of this section) – an example video is shown in Figure 12. The video was created using the files camera-anim-conicspiral.pov and camera-anim-conicspiral.ini. With a lower graphical quality but in realtime a similar camera animation can be simulated in MuPAD using the file camera-anim-conicspiral-MuPAD.mn.

Figure 11: Conical spiral (Video)

Figure 12: Camera animation on a conical spiral

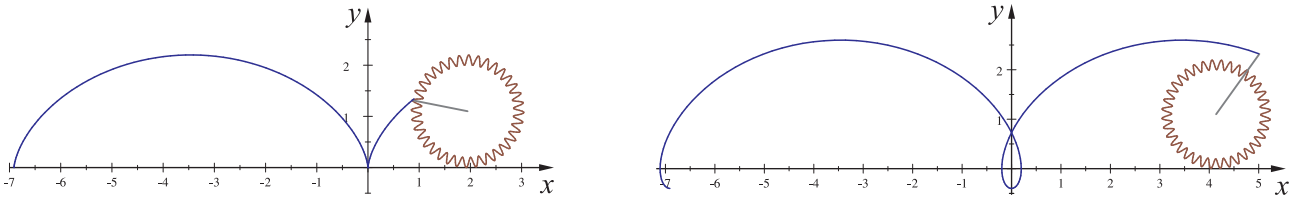
Other variations of the discussed curves emerge by using nonlinear function terms in t for the height or the radius. In the section 4 some results are shown, that students obtained by various functional considerations.

3.4 Cycloids

The motion track of a point (e. g. a bicycle valve) on a circular wheel as the wheel rolls along a straight line can be described by adding the linear motion (translation) of the centre of the circle and the rotation of the point around the centre. In order that these movements are synchronized, the amount of translation has to be $2\pi r$ while the wheel (with radius r) rotates one time. If t is the translation amount (which can be used as parameter) then for the corresponding turning angle φ (in radians) follows $\varphi(t) = \frac{t}{r}$. Therefore the motion track of a point on the circumference of the turning circle can be described by the parametric equation

$$x(t) \;=\; t + r \cdot \sin\left(\tfrac{t}{r}\right); \qquad y(t) \;=\; r + r \cdot \cos\left(\tfrac{t}{r}\right).$$

Figure 13 a) shows a snapshot of an animation of a rolling wheel which was created with the CAS MuPAD. The curve which is described by a point on the wheel is called *cycloid*. Not necessarily a point on the circumference has to be traced, cycloids can also be generated by points in- or outside of a circle, see figure 13 b). If a point has the distance a from the centre, the parametric equation of the corresponding cycloid is



a) b)

Figure 13: Cycloids

Figure 14: Cycloid (Video)

$$x(t) \;=\; t + a \cdot \sin\left(\tfrac{t}{r}\right); \qquad y(t) \;=\; r + a \cdot \cos\left(\tfrac{t}{r}\right).$$

Figure 14 (with $a < r$) shows an animation which was created with POV-Ray (files: cycloid.pov and cycloid.ini). A rolling gear with a small cylinder as "moving point" was generated as follows:

```
union{ object{ Gear(60,1,0.1) pigment {color rgb <0,0,1> } }
       cylinder{ <0,-0.3,-0.25> <0,-0.3,0.25> 0.025
                 texture{Silver_Texture}           }
       rotate <0,0,-360*clock>
       translate <clock*pi,0.5,0>
     }
```

The cycloid which is "drawn" by the moving point consists of a multitude of small spheres:

```
#declare r=0.5;
#declare a=0.3;
#declare i=-2000;
  #while (i <= 2000*clock)
    #declare pt = 2*pi*(i/2000);
    sphere{ <r*pt -a*sin(pt),            // x(t)    parametric
             r    -a*cos(pt),            // y(t)    equation
                   -0.2      >           // z(t)    of the motion track
           0.02 texture{Gold_Texture} } // radius of the small spheres
  #declare i=i+1;
  #end
```

Using a similar approach as for cycloids (rolling a smaller circle outside or inside the circumference of a larger circle) *epicycloids* and *hypocycloids* can be described. Figure 15 shows an epicycloid which is described by the parametric equation

$$x(t) \;=\; (r_1{+}r_2)\cdot\cos(t) + a\cdot\cos\left(t\cdot\left(\tfrac{r_2}{r_1}{+}1\right)\right); \quad y(t) \;=\; (r_1{+}r_2)\cdot\sin(t) + a\cdot\sin\left(t\cdot\left(\tfrac{r_2}{r_1}{+}1\right)\right)$$

with the minor radius r_1 which is a third of the major radius r_2.

- POV-Ray-files for creating animations as shown in Figure 15: epicycloid.pov, epicycloid.ini
- MuPAD-File containing similar animations: epicycloids.mn

Figure 15: Epicycloid

4 Teaching experiences and students results

In a 3 week-project 12th grade high school students of a basic course in mathematics created computer graphics and animations using POV-Ray (see [Filler/Rieper, 2004]). Some of the students decided to choose animations as main topic of their project work. In Figure 16 are shown two example snapshots of animations where students moved target points of spotlights along curves. One student animated a spotlight target along a circle (Fig. 16 a) with the following POV-Ray-Code:

```
light_source{ <20,60,15>  color White
              spotlight  radius 10  falloff 20  tightness 5
              point_at
                <80*sin((3.14*clock)/180), 1, 80*cos((3.14*clock)/180)> }
```

A more sophisticated example was created by another student, who wanted to produce a stunning introduction with a moving illumination of a movie title. After some of experiments with different parametric descriptions of curves she discovered a Lissajous-Curve as a good path for the spotlight target, see Fig. 16 b. Therefore she developed the parametric description

$$\begin{aligned} x(t) &= 150 \cdot \sin \tfrac{t}{15} \\ y(t) &= 40 \cdot \cos \tfrac{t}{30} - 20 \quad t \in [0;300] \\ z(t) &= 60 \end{aligned}$$

and used it in the following part of a POV-Ray-scene description:

```
light_source{ <0,100,-150>  color Orange
              spotlight  radius 8  falloff 9  tightness 1
              point_at <150*sin (clock/15), -20+40*cos(clock/30), 60> }
```

It has to be mentioned that the student who created this animation wasn't a student with special interests in mathematics (as mentioned before she attended a basic course in mathematics). But she was very motivated by the ambition to create an interesting animation and therefore she spent a lot of time thinking about the description of an attractive animation path.

a) b)

Figure 16: Animations, created by high-school-students

In a summer school with 16-18 years old students with special interests for mathematics participants described various curves by parametric equations and explored their properties. After discussing circles, spirals and helices and an introduction into the graphics and animations capabilities of the CAS MuPAD the students were prompted to create beautiful curves by parametric descriptions. Therefore they had to think about adequate coordinate functions to obtain interesting results. Via functional considerations and experiments they created surprisingly fancy shapes. They developed, among others, the following parametric descriptions and created the related graphics The names of the curves were given them by the students.

”Flower curve”

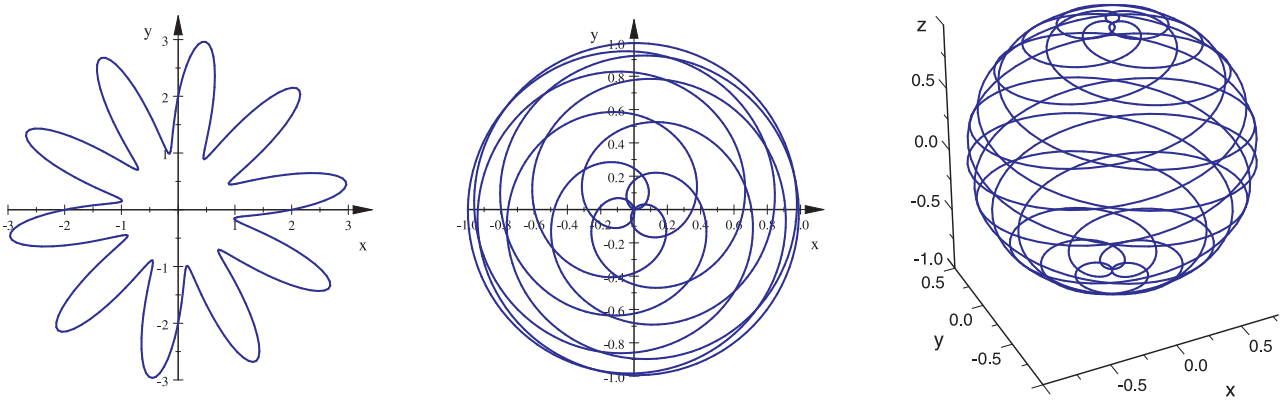
$$\begin{aligned} r(t) &= 2 + \sin(20\pi t) \\ x(t) &= r(t) \cdot \cos(2\pi t) \\ y(t) &= r(t) \cdot \sin(2\pi t) \end{aligned}$$

”Sine circle”

$$\begin{aligned} r(t) &= \sin(t) \\ x(t) &= r(t) \cdot \cos(2\pi t) \\ y(t) &= r(t) \cdot \sin(2\pi t) \end{aligned}$$

”Ball curve”

$$\begin{aligned} r(t) &= \sin(t) \\ x(t) &= r(t) \cdot \cos(2\pi t) \\ y(t) &= r(t) \cdot \sin(2\pi t) \\ z(t) &= \cos(t) \end{aligned}$$



”Flower curve” ”Sine circle” ”Ball curve”

Figure 17: Curves created and named by students

Figure 18: "3D Flower" - Animation, created by high-school-students

An outstanding animation which was created by two students is shown in Figure 18. They started with the two-dimensional "flower curve" (Fig. 17) and after some considerations and experiments they found the following parametric description, which they used to create the curve in Figure 18:

$$\begin{aligned} x(t) &= \left(1+\left(5+\tfrac{7}{2}\cdot\sin(20\pi\cdot t)\right)\cdot t\right)\cdot\cos(2\pi\cdot t) \\ y(t) &= \left(1+\left(5+\tfrac{7}{2}\cdot\sin(20\pi\cdot t)\right)\cdot t\right)\cdot\sin(2\pi\cdot t) \\ z(t) &= \tfrac{1}{4}\,t^2\cdot(1+\sin(20\pi\cdot t)) \end{aligned}$$

The Animation was generated in MuPAD by inserting an additional parameter (here called u) which is recognized as animation parameter:

```
plot::Curve3d( [x(t),y(t),z(t)], t=0..u, u=0..10, Frames=400,
               LineColorType = Rainbow,
               LineColor = RGB::Blue, LineColor2 = RGB::Red)
```

5 Conclusion

The above examples show that it is possible to create interesting curves and animations based on them with simple elementary mathematical tools which tie up with the secondary level I lessons. However, the time needed to do this must not be underestimated. Even for seemingly simple functional considerations which, say, lead from circles to spirals or helices, many students needed quite a lot of time. As they found these considerations interesting, they fell into the joy of experimentation and were

willing to spend a relatively large amount of free time on them. Especially the creation of interesting videos is, as already mentioned, a motivating endeavour for many school students.

Through variations on parametric equations of curves and the resulting description of "new" curves, we get rich possibilities for functional considerations where students – based on qualitative descriptions of desired curve progressions – ponder the function terms through which these can arise and check their considerations using software. From the standpoint of teaching mathematics is especially worthy of mention that these activities triggers sophisticated considerations about functional relationships which often inadequately appear in the current dominant treatment of parametric equations in analytic geometry lessons.

The decision which software to use in lessons should be made considering the further topics of instruction. Both POV-Ray and various CAS provide possibilities to create graphics and animations as described in the following sections. If the exploration of curves is intended to be the main aspect of the following lessons, then using a CAS has some advantages because of its interactive viewing capabilities. If applications of analytic geometry for computer graphics are a main topic, POV-Ray should be preferred, see [Filler/Rieper, 2004], [Filler, 2007] and [Krumpe, 2005]. Generally the experience was made, that using POV-Ray is (because of its photo-realistic graphics quality) very motivating even for students without special interests for mathematics, while students with a preference for deeper mathematical explorations prefer CAS because of the fact that results are often faster accessible.

References

[Filler/Rieper, 2004] Filler, A. and Rieper, F., "3D Computer Graphics and Analytic Geometry in Mathematics Education in Grammar Schools", In: Rogerson, A. (Ed.): Proceedings of the International Conference The Future of Mathematics Education, Ciechocinek, Poland, 2004, 39-44.

[Filler, 2007] Filler, A., *Einbeziehung von Elementen der 3D-Computergrafik in den Mathematikunterricht der Sekundarstufe II* (Habilitationsschrift). e-doc Dokumenten- und Publikationsserver der Humboldt-Universitt zu Berlin: http://dochost.rz.hu-berlin.de, 2007.

[Krumpe, 2005] Krumpe, N., "Creating Three-Dimensional Scenes", *Mathematics Teacher*, 2005, **98**, 6, 394-398.

[Majewski, 2004] Majewski, M., *MuPAD Pro Computing Essentials*, Springer, 2004 (2nd ed.).

[Wells et al., 1993] Wells, D., Young, C. and Farmer, D., *Ray Tracing Creations*, The Waite Group Press, 1993.

[Website Filler] Website with materials regarding this article: 3D computer graphics and the mathematics behind it: http://www.afiller.de/3dcg (last access: October 30th, 2011).

[Website Lohmueller] Website of Friedrich A. Lohmueller with POV-Ray-related tutorials and macros: http://www.f-lohmueller.de/pov_tut/pov__eng.htm (last access: October 30th, 2011).

Software Packages

MuPAD MuPAD, CAS, Part of the MATLAB Symbolic Math Toolbox, a product of MathWorks: http://www.mathworks.com/products/symbolic.

POV-Ray The Persistence of Vision Raytracer, Photo-realistic 3D graphics software (Freeware): http://www.povray.org/.

pjbmp2avi Tool for converting image sequences into video files (Freeware): http://www.mathematik.hu-berlin.de/~filler/3D/download/pjbmp2avi.zip.

Supplemental Electronic Materials

[1] Filler, A.: Extension (include) file for the simple representation of objects of analytic geometry using POV-Ray: https://php.radford.edu/ ejmt/v6n1p2/anageoL.inc.

[2] Filler, A.: POV-Ray file for creating single points on lines given by parametric equations: https://php.radford.edu/ ejmt/v6n1p2//pareq-line1.pov.

[3] Filler, A.: POV-Ray file using while-loops for creating large numbers of points on lines given by parametric equations: http://www.math.hu-berlin.de/~filler/publikat/ejmt/pareq-line2-while.pov.

[4] Filler, A.: MuPAD file with a procedure for creating large numbers of points on lines given by parametric eq.: https://php.radford.edu/ ejmt/v6n1p2/pareq-line-mupad.mn.

[5] Filler, A.: POV-Ray scene file and POV-Ray initialization file for creating a simple animation: https://php.radford.edu/ ejmt/v6n1p2/animation1.pov and https://php.radford.edu/ ejmt/v6n1p2/animation1.ini.

[6] Filler, A.: MuPAD file for creating a simple animation: https://php.radford.edu/ ejmt/v6n1p2/animation1.mn.

[7] Filler, A.: POV-Ray scene file and POV-Ray initialization file for animating uniform and accelerated motions: https://php.radford.edu/ ejmt/v6n1p2/anim-lines.pov and https://php.radford.edu/ ejmt/v6n1p2/anim-lines.ini.

[8] Filler, A.: POV-Ray scene file and POV-Ray initialization file for animating trajectory parabolas: https://php.radford.edu/ ejmt/v6n1p2/trajparabola.pov and https://php.radford.edu/ ejmt/v6n1p2/trajparabola.ini.

[9] Filler, A.: MuPAD file for animating trajectory parabolas: https://php.radford.edu/ ejmt/v6n1p2/trajparabola.mn.

[10] Filler, A.: POV-Ray scene file and POV-Ray initialization file for moving a sphere around a circle: https://php.radford.edu/ ejmt/v6n1p2/anim-circle.pov and https://php.radford.edu/ ejmt/v6n1p2/anim-circle.ini.

[11] Filler, A.: POV-Ray scene file and POV-Ray initialization file for generating a camera animation on a circle: https://php.radford.edu/ ejmt/v6n1p2/camera-anim-circle.pov and https://php.radford.edu/ ejmt/v6n1p2/camera-anim-circle.ini.

[12] Filler, A.: MuPAD file for generating a camera animation on a circle: https://php.radford.edu/ ejmt/v6n1p2/camera-anim-circle-MuPAD.mn.

[13] Filler, A.: MuPAD file for moving a sphere on an Archimedes spiral: https://php.radford.edu/ ejmt/v6n1p2/anim-spiral-MuPAD.mn.

[14] Filler, A.: MuPAD file for moving a sphere on a helix: https://php.radford.edu/ ejmt/v6n1p2/anim-helix-MuPAD.mn.

[15] Filler, A.: POV-Ray scene file and POV-Ray initialization file for creating an animated conic spiral: https://php.radford.edu/ ejmt/v6n1p2/anim-conicspiral.pov and https://php.radford.edu/ ejmt/v6n1p2/anim-conicspiral.ini.

[16] Filler, A.: MuPAD file for creating an animated conic spiral: https://php.radford.edu/ ejmt/v6n1p2/anim-conicspiral-MuPAD.mn.

[17] Filler, A.: POV-Ray scene and initialization file for generating a camera animation on a conic spiral: https://php.radford.edu/ ejmt/v6n1p2/camera-anim-conicspiral.pov and https://php.radford.edu/ ejmt/v6n1p2/camera-anim-conicspiral.ini.

[18] Filler, A.: MuPAD file for generating a camera animation on a conic spiral: https://php.radford.edu/ ejmt/v6n1p2/camera-anim-conicspiral-MuPAD.mn.

[19] Filler, A.: POV-Ray scene and initialization file for rolling a gear on a plane and generating a cycloid: https://php.radford.edu/ ejmt/v6n1p2/cycloid.pov and https://php.radford.edu/ ejmt/v6n1p2/cycloid.ini.

[20] Filler, A.: MuPAD file for rolling a gear on a line and generating cycloids: https://php.radford.edu/ ejmt/v6n1p2/cycloids.mn.

[21] Filler, A.: POV-Ray scene and initialization file for rolling a gear around another gear and generating a epicycloid: https://php.radford.edu/ ejmt/v6n1p2/epicycloid.pov and https://php.radford.edu/ ejmt/v6n1p2/epicycloid.ini.

[22] Filler, A.: MuPAD file for rolling a gear around another gear and generating epicycloids: https://php.radford.edu/ ejmt/v6n1p2/epicycloids.mn.