Inhaltsverzeichnis

Meine Matura-Arbeit

Abstract

Diese Matura-Arbeit soll einen kleinen Einblick in unser grosses Sonnensystem geben. Wie schon erahnt ist es eine Matura-Arbeit im naturwissenschaftlichen Bereich, genauer in der Astronomie. Der praktische Teil der Arbeit war, im Gegensatz zum schriftlichen Teil, viel mehr in Richtung Informatik, denn der praktische Teil dabei war, das Programm zu schreiben, welches das Sonnensystem aufzeigen soll mit den verschiedenen Planeten und Planetenbahnen. Im schriftlichen Teil wird hier dem Leser der theoretische Teil nähergebracht.

Diese Matura-Arbeit beanspruchte Mathematik und Physik und gab einen kleinen Einblick in die Astronomie und die Komplexität des Ganzen.

Vorwort

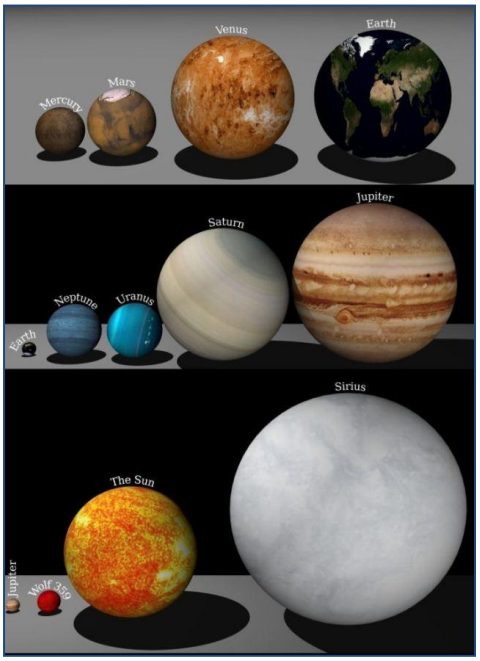
Als Matura-Arbeit war mir klar, dass ich ein Programm schreiben wollte, das Thema aber liess anfangs noch zu wünschen übrig. Ein Stundenplanerstellungsprogramm schien mir als eine gute Wahl, jedoch ergab sich, dass dies zu schwer für einen Mittelschüler ist. Danach war ich der Meinung, dass ich ein Programm schreiben könnte, welches Sonnenfinsternisse berechnet. Als ich dies dann meinem Schwerpunktfachlehrer in Mathematik Hr. Dipl. Math. Markus Leisibach vorschlug, sagte er mir, er hätte mir dasselbe auch schon vorschlagen wollen. Somit waren keine Zweifel mehr für das Thema meiner Matura-Arbeit vorhanden.

Mit der Zeit begann sich das Ziel Sonnenfinsternisse zu berechnen zu verabschieden, da es zu schwer eingestuft wurde, was meine Matura-Arbeit zu einer Verwandlung zwängte. Die Berechnungen der Sonnenfinsternisse setzte die Mond-Bahn-Berechnung voraus, welche von der Sonne und der Erde beeinflusst werden. Dieses Problem kennt man unter dem Namen „Drei-Körper-Problem“ welches für einen Mittelschüler eine Sache der Unmöglichkeit darstellt. Somit wurde aus der Berechnung von Sonnenfinsternissen die Berechnung der Planetenumlaufbahnen.

Die Astronomie war mir als Kind ein relativ wichtiger Punkt. Ich war fasziniert von Astronauten und Raketen. Ich hatte sogar schon Bücher über unser Sonnensystem, insbesondere dieses, welches die Rakete „Saturn“ aufzeigte, was nebenbei auch der Name des von mir meistgeliebten Planeten darstellte. Diese Matura-Arbeit erinnerte mich so ein Weilchen an meine Kindheit.

Ich möchte mich natürlich noch bei denjenigen bedanken, ohne die ich diese Arbeit nicht hätte beenden können: Hr. Dipl. Math. Markus Leisibach, mein Schwerpunktfach -Mathematik-Lehrer und Betreuer, Hr. Dipl. Phys. Peter Strickler, mein Mathematik-, Physik- und Schwerpunktfach-Physik-Lehrer und Mário Correia Sampaio, meinen Vater, der mich bei der Arbeit ständig betreut und begleitet hat.

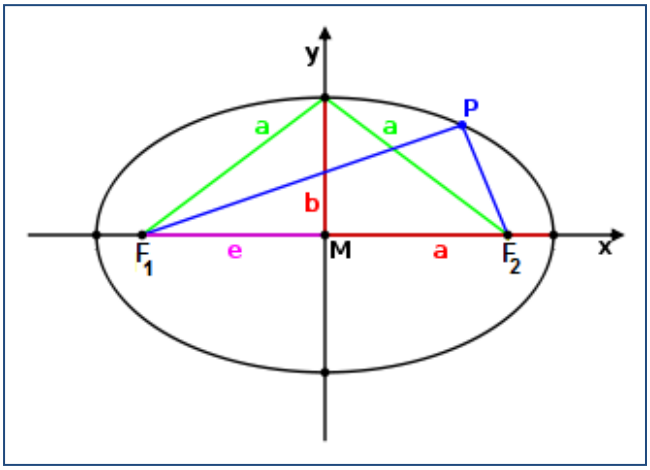
Einleitung



Bilder sagen mehr als tausend Worte, nicht wahr? Allein unsere Erde ist schon von Riesenmassen umzingelt. Wir sind in unserem Sonnensystem ein winziges kleines Ding und doch sind wir nicht unbedeutend. Unsere kleine Erde ist bisher der einzig bekannte Ort indem Leben herrschen kann. Ist es durch Zufall, ist es von Gottes Hand geschaffen, niemand weiss es, aber jeder will es wissen. Diese Willenskraft macht aus uns Menschen etwas Besonderes.

Mit dieser Matura-Arbeit sind wir unserem Sonnensystem ein wenig näher. Meine praktische Arbeit befasst sich mit unseren acht Planeten und ihren Bahnen. Ich kreierte ein Programm, welches eine 2-dimensionale Darstellung unseres Sonnensystems in der Vogelperspektive aufzeigt. Meine schriftliche Arbeit hier befasst sich mit der Theorie des Ganzen und der Basis des Programms.

Die Ellipse



Variablen und ihre Bedeutungen:

a = grosse Halbachse

b = kleine Halbachse

e = lineare Exzentrizität
(Abstand zwischen Mittel- und Brennpunkt)

$F_{1,2}$ = Brennpunkte

M = Mittelpunkt

Die wichtigsten Formeln der Ellipse

Ellipsengleichung:

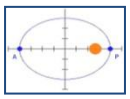
$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1 \quad oder \quad y^2=b^2\cdot\left(1-\frac{x^2}{a^2}\right)$$

$$e=\sqrt{a^2-b^2}=a\cdot(1-\varepsilon)$$

ε = numerische Exzentrizität
(eine Zahl zwischen 1 und 0, welche angibt, wie stark die Ellipse von einem Kreis abweicht)

Die Keplerschen Gesetze

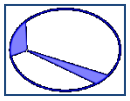
1. Keplersches Gesetz



Die Planeten bewegen sich auf elliptischen Bahnen in deren Brennpunkt die Sonne liegt. (Der Punkt der der Sonne am nächsten liegt heisst Perihel, der am weitesten Aphel.)

Diese Tatsache verändert die Ellipsengleichung wie folgt: $y^2=b^2\cdot\left(1-\frac{(x+e)^2}{a^2}\right)$

2. Keplersches Gesetz



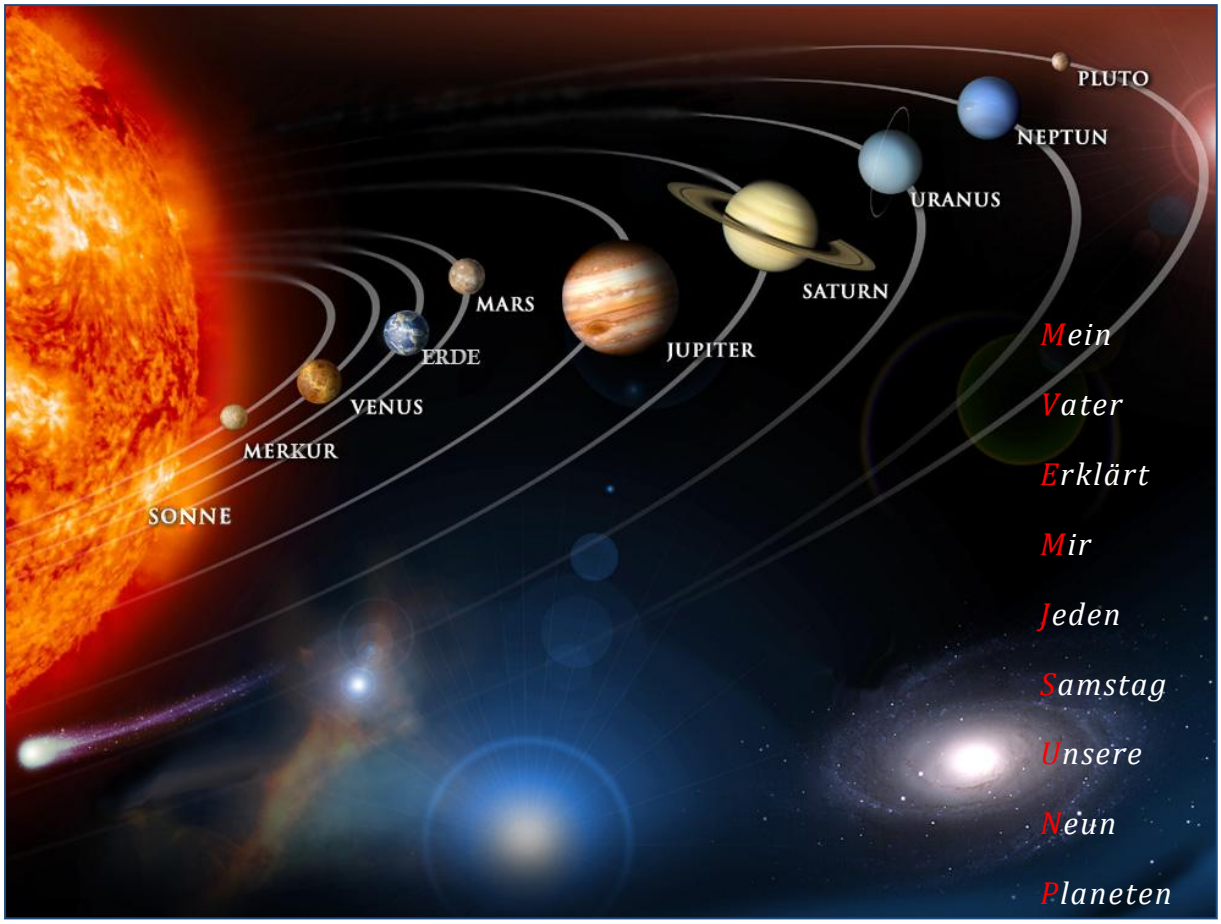
Ein von der Sonne zum Planeten gezogener „Fahrstrahl“ überstreicht in gleichen Zeiten gleich große Flächen.

3. Keplersches Gesetz

Die zweiten Potenzen der Umlaufzeiten zweier Planeten verhalten sich wie die dritten Potenzen der grossen Halbachsen.

$$\frac{{T_1}^2}{{a_1}^3}=\frac{{T_2}^2}{{a_2}^3}$$

Unsere acht Planeten



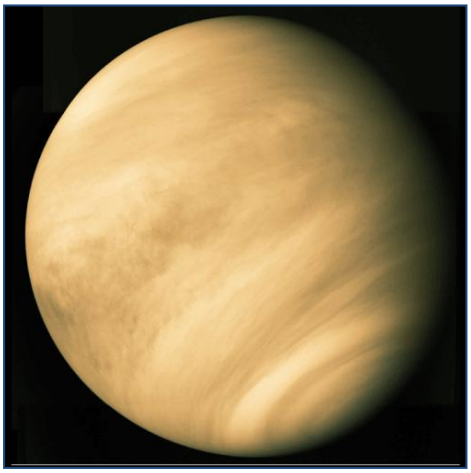
Wer kennt dieses alte Sprichwort schon nicht. Mein Vater erklärt mir jeden Samstag unsere neun Planeten. Seitdem Pluto nicht mehr zu den Planeten, sondern zu den Zwergplaneten gehört, kann man es jedoch nicht mehr sinngemäss anwenden. Ohne Pluto sind es nur noch acht Planeten, mit denen man bis heute noch keinen Spruch erfunden hat.

Merkur ☿

Merkur ist mit einem durchschnittlichen Sonnenabstand von knapp 58 Mio. Kilometern der sonnennächste Planet im Sonnensystem. Im Vergleich mit anderen Planeten, weicht die Bahn des Merkurs am meisten von einem Kreis ab, mit einer numerischen Exzentrizität von ungefähr 0,2.

$Masse\ (m) = 3.3022 \cdot 10^{23} kg$
$Grosse\ Halbachse\ (a) = 57'909'175'000m$
$numerische\ Exzentrizität\ (\varepsilon) = 0.20563069$

Venus ♀



Venus ist der zweitinnerste Planet im Sonnensystem mit einer durchschnittlichen Sonnenentfernung von 108 Mio. km. Die Venus ist nach dem Mond das hellste natürliche Objekt an unserem Himmel. Sie erscheint morgens und abends, weshalb sie auch Morgenstern bzw. Abendstern genannt wird.

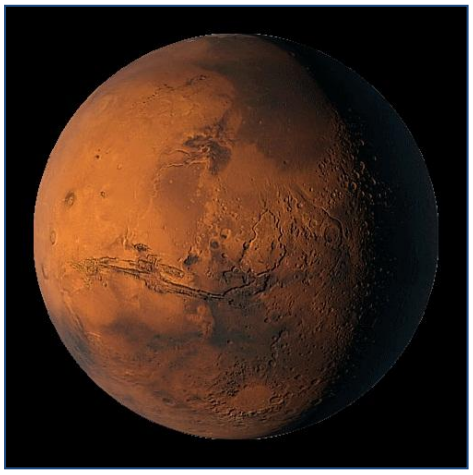
$Masse\ (m) = 4.8685 \cdot 10^{24} kg$
$Grosse\ Halbachse\ (a) = 108'208'930'000m$
$numerische\ Exzentrizität\ (\varepsilon) = 0.00677323$

Erde ♁

Unsere Erde umkreist die Sonne mit einer durchschnittlichen Entfernung von 150 Mio. km. Bis jetzt ist die Erde der einzig bekannte Planet auf dem Leben herrscht. Momentan hat die Erde eine Bevölkerung von rund 7 Milliarden Menschen.

$Masse\ (m) = 5.9737 \cdot 10^{24} kg$
$Grosse\ Halbachse\ (a) = 149'597'890'000m$
$numerische\ Exzentrizität\ (\varepsilon) = 0.01671022$

Mars ♂



Mars ist der vierte Planet (von der Sonne aus gesehen) mit einem Abstand von knapp 228 Mio. km. Er ist der äussere Nachbar unseres Planeten Erde. Im Vergleich zur Erde ist er aber deutlich kleiner, denn der Durchmesser des Mars' ist geradeмal halb so gross wie der der Erde.

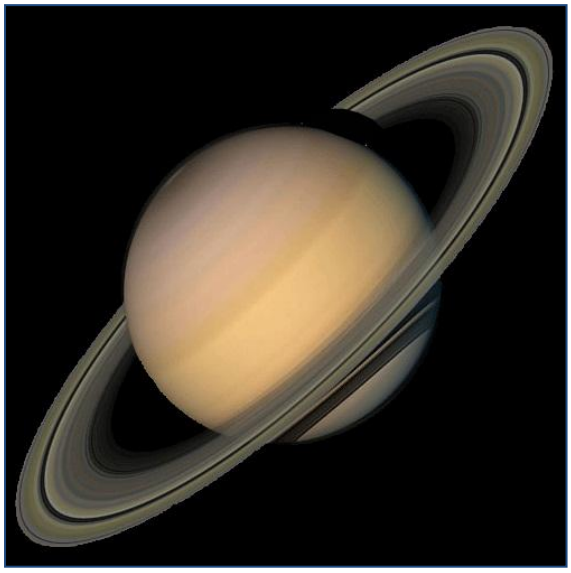
$Masse\ (m) = 6.4185 \cdot 10^{23} kg$
$Grosse\ Halbachse\ (a) = 227'936'640'000m$
$numerische\ Exzentrizität\ (\varepsilon) = 0.09341233$

Jupiter ♃

Jupiter ist der fünfte Planet im Sonnensystem mit einer Entfernung von der Sonne, die durchschnittlich 778 Mio. km beträgt. Mit fast zwei Quadrilliarden kg ist Jupiter der grösste Planet im Sonnensystem. Was man den Zahlen auch erkennen kann, ist, dass man zwischen Mars und Jupiter den ersten Riesenabstand entdeckt, nämlich von 550 Mio. km (Das sind mehr als 3½ Erd-Sonnen-Abstände!). Dieser Abstand lässt sich aber einfach begründen: Zwischen diesen beiden Planeten befindet sich der Asteroiden-Gürtel, der die terrestrischen (erdähnlichen) Planeten von den Gasplaneten trennt.

$$Masse\ (m) = 1.8987 \cdot 10^{27} kg$$
$$Grosse\ Halbachse\ (a) = 778'412'020'000 m$$
$$numerische\ Exzentrizität\ (\varepsilon) = 0.04839266$$

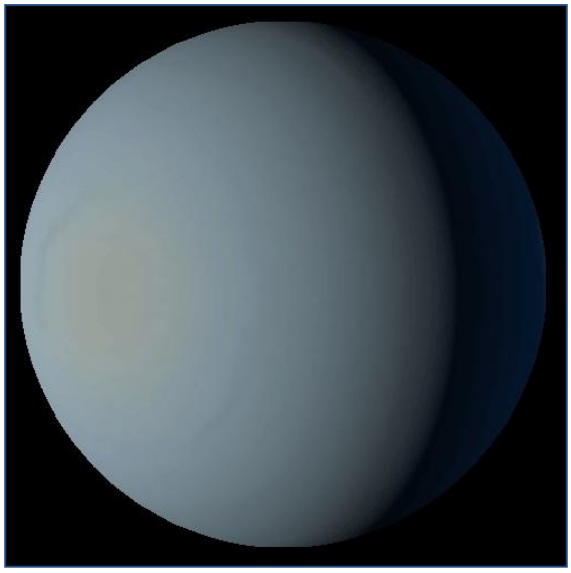
Saturn ♄



Saturn ist der sechste Planet im Sonnensystem mit einem durchschnittlichen Abstand von 1,4 Mia. km. Der Saturn ist in meiner Sichtweise der schönste Planet (von weitem betrachtet), weshalb ich auch empfinde, dass er mit seinen über 100‘000, in solch schöner Weise angereihten, Ringen unser komplettes Sonnensystem verschönert. Aber nicht alles was glänzt ist Gold und so auch hier: Von aussen gesehen ist der Planet zwar wunderschön, aber auf dem Planeten selbst spielen sich die heftigsten Gewitter ab, die je gesichtet wurden.

$$Masse\ (m) = 5.6851 \cdot 10^{26} kg$$
$$Grosse\ Halbachse\ (a) = 1'426'725'400'000 m$$
$$numerische\ Exzentrizität\ (\varepsilon) = 0.05415060$$

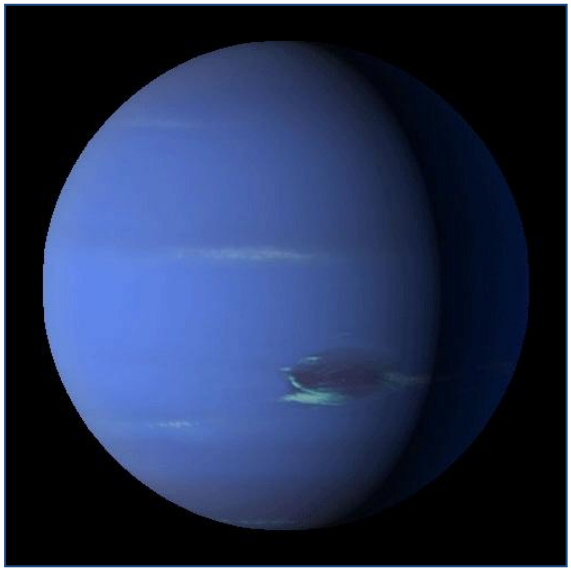
Uranus ♅



Uranus, der siebte Planet, hat eine Entfernung von mehr als 2,8 Mia. km von der Sonne. Er ist, wie Jupiter und Saturn einer der Gasplaneten. Zusammen mit dem Planeten Neptun bildet Uranus eine Gruppe von Planeten mit dem Namen „Eisriesen“, da eine Eisschicht in ihrem Innern existiert.

$Masse\ (m) = 8.6849 \cdot 10^{25} kg$
$Grosse\ Halbachse\ (a) = 2'870'972'200'000 m$
$numerische\ Exzentrizität\ (\varepsilon) = 0.04716771$

Neptun ♆



Last but not least: Der Planet Neptun. Er bildet sozusagen das „Schlusslicht“ der Planeten unseres Sonnensystems mit einer Durchschnittsentfernung von 4½ Mia. km. Neptun ist auch der Planet mit den am weitesten entfernten Nachbarn. Der Planet Uranus ist beinahe 1,7 Mia. km entfernt und auch der nachfolgende Zwergplanet Pluto ist nicht gerade nahe bei ihm, denn der ist auch noch 1,4 Mia. km entfernt von Neptun.

$Masse\ (m) = 1.0244 \cdot 10^{26} kg$
$Grosse\ Halbachse\ (a) = 4'498'252'900'000 m$
$numerische\ Exzentrizität\ (\varepsilon) = 0.00858587$

Definition eines Planeten: *„Ein Planet ein Himmelskörper, der sich auf einer Umlaufbahn um die Sonne bewegt, dessen Masse groß genug ist, dass sich das Objekt im hydrostatischen Gleichgewicht befindet – und somit eine näherungsweise kugelähnliche Gestalt besitzt – und der das dominierende Objekt seiner Umlaufbahn ist, das heißt, diese über die Zeit durch sein Gravitationsfeld von weiteren Objekten „geräumt“ hat.“*

Oder einfach gesagt: Ein Planet ist dann ein Planet, wenn er eine Umlaufbahn um einen Stern hat, kugelförmig ist und seine Bahn von niemand anderem gelenkt wird, als von ihm.

Berechnung der Bahnpunkte

Theorie

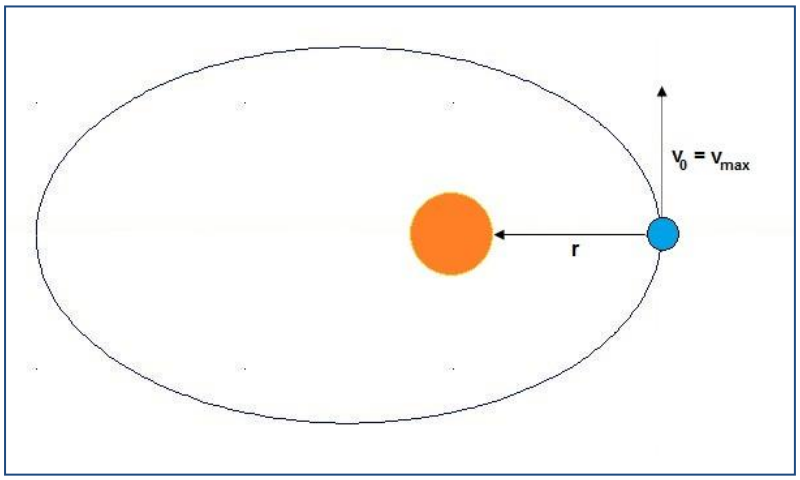
Was man zuerst noch wissen sollte

Aus unserem Mathematik- und Physikunterricht kennen wir Gesetze, welche uns ermöglichen viele Dinge in unserem Lebensraum zu berechnen. So kann man z.B. voraussagen, mit welcher Geschwindigkeit ein Stein von einem 80-Meter-Turm am Boden ankommen wird oder ob ein Auto noch rechtzeitig halten wird und so einen Unfall vermeiden kann oder eben nicht. Auch wenn es nicht so erscheint, sind eben diese erwähnten Sachen die Grundlage von dem, was meine Arbeit beinhaltet. Es sind die physikalischen Gesetze der Kinematik (die Gesetze der gleichförmigen Bewegung) und die Gesetze der Gravitation (die Gesetze der Schwerkraft):

$$F = m \cdot a \qquad F_G = G \cdot \frac{m_1 \cdot m_2}{r^2} \qquad s = v \cdot \Delta t = \frac{1}{2} \cdot a \cdot t^2 \qquad v = a \cdot \Delta t = \sqrt{2 \cdot a \cdot s}$$

Aus diesen paar Gleichungen kann man mittels Computer jeden Punkt der Planetenbahnen numerisch berechnen. So wie sie jetzt aber dastehen, kann man noch nichts damit anfangen. Man muss diese Gleichungen zuerst noch nach einem gewissen Schema ordnen und umformen.

Gehen wir zuerst einmal von der Grundsituation aus, dass der Planet sich im Perihel befindet, also am sonnennächsten Punkt:

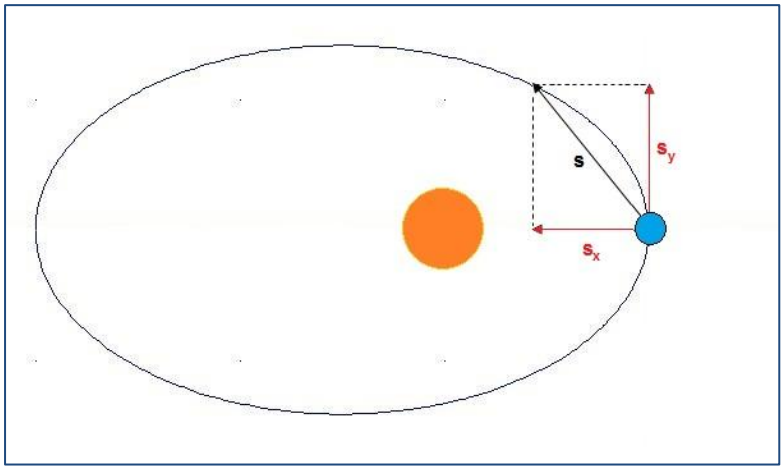


Umso näher der Planet an der Sonne liegt, umso grösser ist die Anziehungskraft auf ihn und so ist es auch mit der Geschwindigkeit der Planeten: Umso näher der Planet an der Sonne liegt, umso grösser seine Geschwindigkeit. Im Perihel ist die Geschwindigkeit also am grössten, wenn wir dieser Überlegung folgen.

Im Physikunterricht wird immer wieder gesagt, dass die Geschwindigkeit in Tangentenrichtung ist und so ist es auch hier beim Planeten. Am Perihel hat der Abstand r und die Geschwindigkeit v_0 einen Zwischenwinkel von 90°. Wenn wir die Geschwindigkeit in x- und in y-Richtung betrachten sehen wir, dass an diesem Punkt $v_x = 0$ ist und $v_y = v_0$ bzw. $v_y = v_{max}$ ist.

Von A nach B kommen

Betrachten wir nun, was das für unseren neuen Punkt bedeutet:



In y-Richtung haben wir die kinetische Energie, die uns in v-Richtung vorwärtsbewegt. Somit ist nach den Gesetzen der Kinematik: $s_t = v_t \cdot \Delta t$.

Das wäre schon unser nächster Punkt, wenn da nicht noch Anziehungskraft der Sonne wäre. Diese zieht den Planeten Richtung Sonne oder anders gesagt: Sie zieht den Planeten in Richtung Zentrum. Um dies zu berechnen haben wir ja glücklicherweise noch eine Formel, die wir auch gerne anwenden, nämlich: $s_t = a \cdot \Delta t^2$.

Die Geschwindigkeit ist hier in y-Richtung. Somit ist das obere $s_t = s_y$ und da die Gravitation der Sonne in x-Richtung zeigt, ist das untere $s_t = s_x$.

Der nächste Punkt ist somit $P_t = \left(s_{x_{t-\Delta t}} + s_x \,/\, s_{y_{t-\Delta t}} + s_y\right)$.

Halbschrittverfahren

Um die Koordinaten des nächsten Punktes zu bekommen sind also Gleichungen nötig in der Form:

$$x_{(t)} = x_{(t-\Delta t)} + v_x \cdot \Delta t \;\; und \;\; y_{(t)} = y_{(t-\Delta t)} + v_y \cdot \Delta t.$$

Das Problem hierbei ist aber die Geschwindigkeit. Wie erwähnt ist die Geschwindigkeit bei jedem Punkt wieder anders. Hier kommt ein Verfahren namens „Halbschrittverfahren“ ins Spiel. Es ist eine Näherung für die Geschwindigkeit an diesem genauen Punkt.

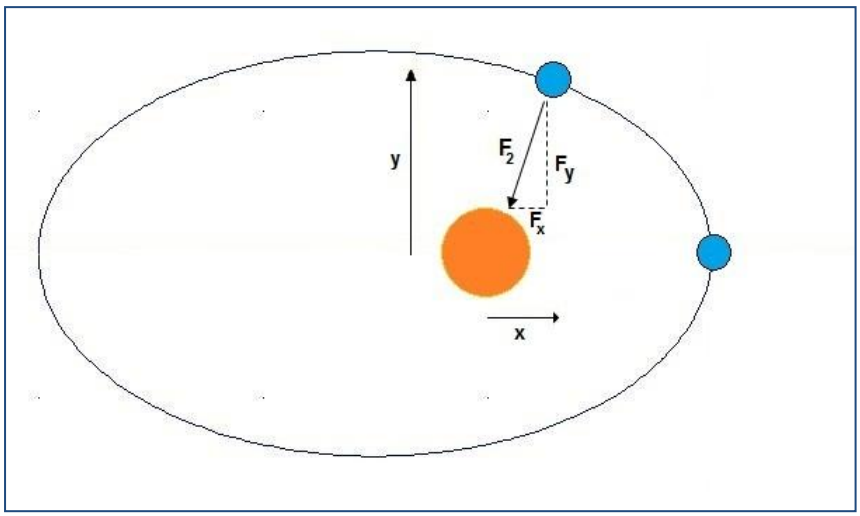
Dies sieht wie folgt aus:

$$x_{(t)} = x_{(t-\Delta t)} + v_{x_{\left(t-\frac{\Delta t}{2}\right)}} \cdot \Delta t \quad bzw. \quad y_{(t)} = y_{(t-\Delta t)} + v_{y_{\left(t-\frac{\Delta t}{2}\right)}} \cdot \Delta t$$

Die Geschwindigkeit wird jetzt mit Hilfe der Kinematik-Gesetze ($v_t = v_{t-\Delta t} + a_t \cdot \Delta t$) ermittelt:

$$v_{x_{\left(t-\frac{\Delta t}{2}\right)}} = v_{x_{\left(t-\frac{3}{2}\Delta t\right)}} + a_{x_{(t)}} \cdot \Delta t \quad bzw. \quad v_{y_{\left(t-\frac{\Delta t}{2}\right)}} = v_{y_{\left(t-\frac{3}{2}\Delta t\right)}} + a_{y_{(t)}} \cdot \Delta t$$

Nun fehlt nur noch die Berechnung der Beschleunigungen in x- und in y-Richtung. Machen wir uns zuerst einmal ein Bild davon:



Auf den Planeten wirkt 1 einzige Kraft, die Gravitationskraft:

$$F_G = -G \cdot \frac{m \cdot M}{r^2}$$

Um nun die Beschleunigungen zu berechnen brauchen wir eine zweite, Formel, nämlich das 1. Newtonsche Gesetz:

$$F = m \cdot a$$

Die Gravitationskraft ist eine Zentralkraft, was heisst, dass die Kraft immer in Richtung Zentrum (=Mitte) geht. Somit ergibt sich die Analogie, dass die Kräfte-Komponenten sich genau wie die Weg-Komponenten verhalten:

$$\frac{F_x}{F_y} = \frac{x}{y}$$

Betrachten Sie zum Verständnis die Abbildung oben.

Das gleiche gilt auch für den Abstand r zwischen Sonne und Planet:

$$\frac{F_x}{F} = \frac{x}{r} \quad bzw. \quad \frac{F_y}{F} = \frac{y}{r}$$

Wenn man nun F durch $F = G \cdot \frac{m \cdot M}{r^2}$ ersetzt und F_x/F_y durch $F_{x/y} = m \cdot a_{x/y}$, durch m teilt und nach a auflöst bekommt man:

$$a_{x_{(t)}} = -G \cdot M \cdot \frac{x_{(t)}}{r_{(t)}^3} \quad bzw. \quad a_{y_{(t)}} = -G \cdot M \cdot \frac{y_{(t)}}{r_{(t)}^3}$$

Da r nichts weiter als $\sqrt{x^2 + y^2}$ ist, ist man damit fertig, bzw. fast fertig, denn das ganze Verfahren hat nur 1 Haken, nämlich $v_{t-\frac{3}{2}\Delta t}$. Für die Berechnung des Starts braucht man die Geschwindigkeit des letzten Punktes. Um dies zu umgehen, setzt man für den ersten Berechnungsschritt, abweichend vom obigen Verfahren, folgendes ein:

$$v_{x_{t-\frac{3}{2}\Delta t}} = v_{x_0} + a_{x_0} \cdot \frac{\Delta t}{2} \quad bzw. \quad v_{y_{t-\frac{3}{2}\Delta t}} = v_{y_0} + a_{y_0} \cdot \frac{\Delta t}{2}$$

Wie es der Computer haben muss

Damit der Computer das auch rechnen kann, muss man dies noch so ordnen, dass es der Planet kontinuierlich rechnen kann, so dass er

1. $a_{x_{t-\Delta t}} = -G \cdot M \cdot \frac{x_{t-\Delta t}}{r^3_{t-\Delta t}}$
2. $a_{y_{t-\Delta t}} = -G \cdot M \cdot \frac{y_{t-\Delta t}}{r^3_{t-\Delta t}}$
3. $v_{x_{t-\frac{\Delta t}{2}}} = v_{x_{t-\frac{3}{2}\cdot\Delta t}} + a_{x_{t-\Delta t}} \cdot \Delta t$
4. $v_{y_{t-\frac{\Delta t}{2}}} = v_{y_{t-\frac{3}{2}\cdot\Delta t}} + a_{y_{t-\Delta t}} \cdot \Delta t$
5. $x_t = x_{t-\Delta t} + v_x \cdot \Delta t$
6. $y_t = y_{t-\Delta t} + v_y \cdot \Delta t$

$$\begin{cases} v_{x_{t-\frac{3}{2}\Delta t}} = v_{x_0} + a_{x_0} \cdot \frac{\Delta t}{2} \\ v_{y_{t-\frac{3}{2}\Delta t}} = v_{y_0} + a_{y_0} \cdot \frac{\Delta t}{2} \end{cases}$$

(für den 1. Schritt)

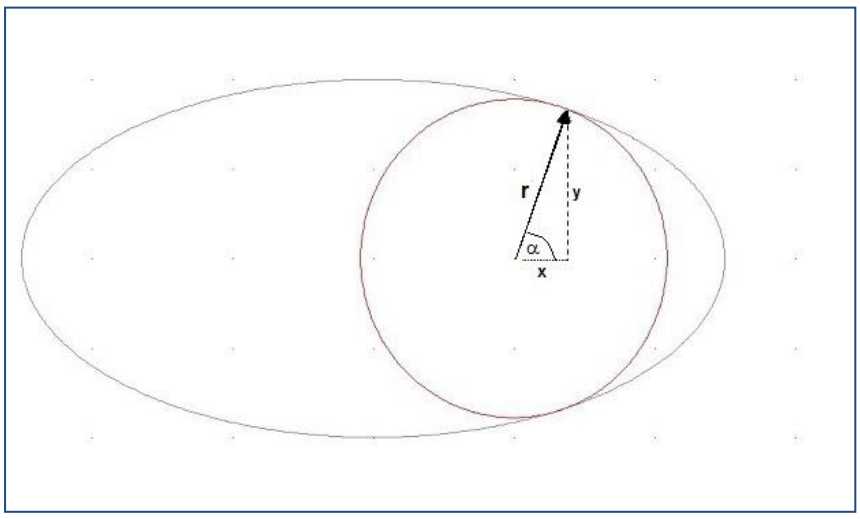
In dieser Reihenfolge kann man ein Programm schreiben, welches immer wieder den nächsten Punkt auf der Bahn berechnet. Damit aber das Programm auch funktioniert muss man einen Startwert haben, d.h. es muss die Startkoordinaten haben sowie auch die Startgeschwindigkeit.

Berechnung der Startkoordinaten

Die sogenannten „Ephemeriden“ der Planeten werden immer in astronomischen Jahrbüchern eingetragen. Sie sagen uns aus, unter welchem Grad α der Planet zur Sonne steht und welchen momentanen Abstand r sie zur Sonne haben. Aus diesen zwei Werten kann man so die Anfangskoordinaten der Planeten bestimmen.

$x = r \cdot \cos(\alpha)$
$y = r \cdot \sin(\alpha)$

Diese zwei Formeln stellen nichts anderes dar als den Schnittpunkt eines Kreises mit der Ellipse. Da man den Winkel und den Abstand weiss, ist der Rest nur noch Trigonometrie:



Berechnung der Startgeschwindigkeit

Bei den Startkoordinaten ist alles noch ganz einfach, aber man muss auch noch die Startgeschwindigkeit haben, damit das Ganze funktioniert. Damit wir diese berechnen können, müssen wir uns die Ellipse nochmals unter die Lupe nehmen. Wie anfangs schon gesagt, geht die Geschwindigkeit immer in Tangentenrichtung weiter. Um die Geschwindigkeit also zu wissen, müssen wir die Tangentensteigung herausfinden, denn es gilt:

$$m_t = \frac{v_y}{v_x} \quad ①$$

Um die Steigung der Tangente zu berechnen, müssen wir wieder zurück an den Anfang, zur Ellipsengleichung. Diese lautet ja wie folgt:

$$f_{(x)} = b \cdot \pm\sqrt{\left(1 - \frac{(x+e)^2}{a^2}\right)}$$

Wie wir im Mathematikunterricht gelernt haben, ist die Ableitung von $f_{(x)}$ die Steigung am Punkt x. Diese Steigung ist zugleich die Tangentensteigung:

$$m_{t_{(x)}} = f'_{(x)} = \left(b \cdot \pm\sqrt{\left(1 - \frac{(x+e)^2}{a^2}\right)}\right)' = \frac{b \cdot (x+e)}{a \cdot \mp\sqrt{a^2 - (x+e)^2}}$$

Seit wir den Satz des Pythagoras kennen, kennen wir auch diese Formel:

$$v = \sqrt{v_x^2 + v_y^2} \quad ②$$

Um v zu berechnen benötigen wir nun noch eine weitere Formel. Dieses Mal kommt sie nicht von der Mathematik oder Physik, sondern aus der Astronomie. Wie auch die Astronomie einen schönen Namen hat, so hat es auch diese Formel in sich mit ihrem Namen. Sie heisst „Vis-Viva-Gleichung“ und sie lautet wie folgt:

$$v^2 = G \cdot M \cdot \left(\frac{2}{r} - \frac{1}{a}\right) \quad bzw. \quad v = \sqrt{G \cdot M \cdot \left(\frac{2}{r} - \frac{1}{a}\right)}$$

Nun da wir die benötigten Formeln haben müssen wir sie noch umformen, damit wir v_x und v_y berechnen können:

① $v_x = \frac{v_y}{m_t}$

② $v_y = \sqrt{v^2 - v_x^2}$

② in ①: $v_x = v \cdot \mp\sqrt{\frac{1}{m_t^2+1}} \quad \rightarrow \quad v_y = \sqrt{v^2 - v_x^2}$

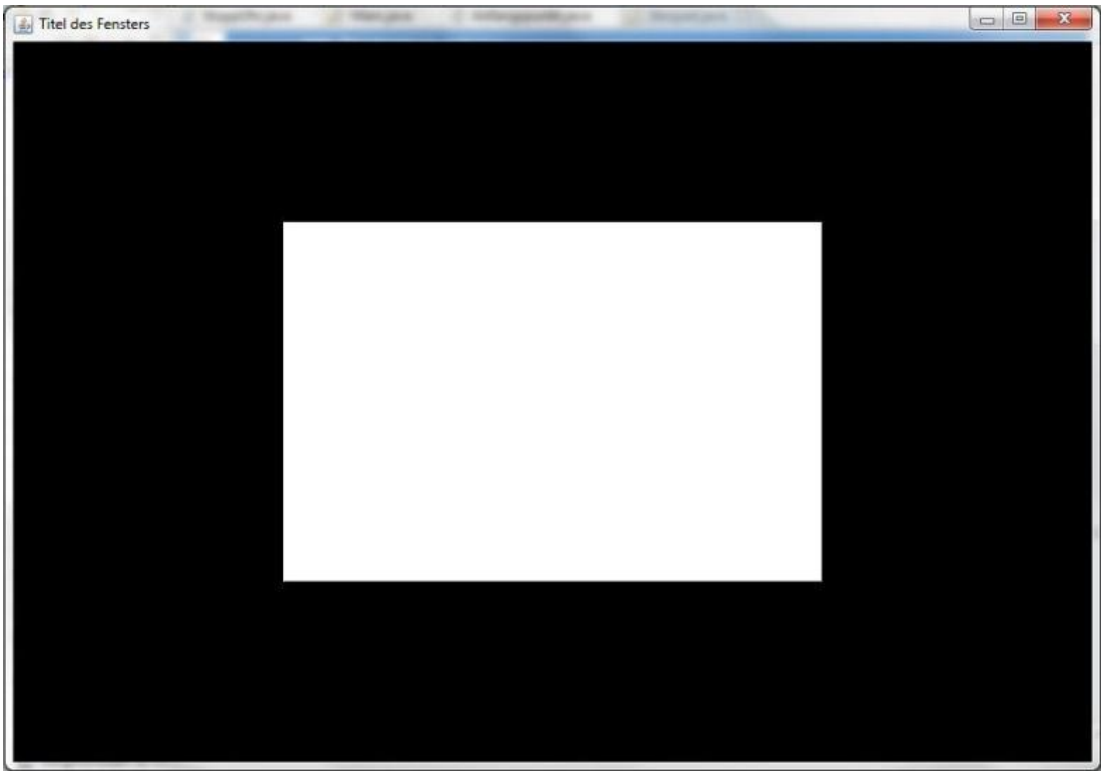
Da die Wurzel negativ oder positiv sein kann, müssen wir uns nun noch überlegen, wann was der Fall ist. Da die Planeten (von oben herab betrachtet) im Gegenuhrzeigersinn wandern, hat der Planet oberhalb der x-Achse eine negative Geschwindigkeit in x-Richtung. Für v_x haben wir es also schnell berechnet. Bei v_x ist also die Wurzel für positive y-Werte negativ und für negative y-Werte positiv. Bei der Geschwindigkeit in x-Richtung waren die Wendepunkte der Perihel und der Aphel. Bei der Geschwindigkeit in y-Richtung ist es ebenfalls so, dass der Wendepunkt ein Extrempunkt der Parabel ist. Es sind die Punkte, wo y der kleinen Halbachse entspricht. Da die Sonne aber nicht im Mittelpunkt der Ellipse, sondern im Brennpunkt, ist x bei diesen Punkten nicht 0, sondern –e(= –a·ε). Somit ist v_y für alle Werte, die kleiner als –e sind negativ und für alle Werte die grösser als –e sind positiv.

Programmierung

Für mein Programm habe ich mich für die Programmiersprache Java entschieden, da ich darin schon Erfahrung habe und auch im Schwerpunktfach Mathematik damit gearbeitet wurde.

Struktur eines Java-Programms

Damit ich zeigen kann, wie die Grundstruktur eines solchen Java-Programms aussieht, zeige ich das an folgendem Beispiel:



Aufgabe

Schreib ein Programm, welches ein ausgefülltes Rechteck, bzw. wenn man das Fenster zusammenzieht ein Quadrat, in die Mitte des Fensters zeichnet. Die Länge des Rechtecks soll gleichlang wie die halbe Fensterlänge und die Höhe des Rechtecks soll gleichgross wie die halbe Fensterhöhe sein.

Lösung

```
/*
 * Welche JAVA-Pakete nötig sind,
 * damit das Programm die Befehle ausführt
 */
import javax.swing.JPanel;
import javax.swing.JFrame;
import java.awt.Color;
import java.awt.Graphics;

public class Beispiel extends JPanel{
	/*
	 * Welche Variablen benötigt werden,
	 * Was diese Variablen darstellen
	 * (ganze Zahlen, reelle Zahlen, ..., Text)
	 */
	static int w,h;

	public static void main (String[] args) {
		/*
		 * Initialisierung der Variablen
		 * (Man kann das auch direkt oben machen)
		 * Starten des Programms
		 */
		w=916;
		h=638;
		new Beispiel().run();
	}

	public void run() {
		JFrame frame = new JFrame("Titel des Fensters");
		frame.setSize(w,h);
		frame.getContentPane().add(this);
		frame.setVisible(true);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		/*
		 * Was das Programm ausführen/rechnen soll
		 */
	}

	public void paintComponent(Graphics g) {
		super.paintComponent(g);
		w = getWidth();
		h = getHeight();
		/*
		 * Was muss in diesem Fenster gezeichnet,
		 * oder geschrieben werden
		 */
		setBackground(Color.black);
		g.setColor(Color.white);
		g.fill3DRect(w/4, h/4, w/2, h/2, true);
	}
}
```

Damit das Rechteck auch in der Mitte des Fensters liegt muss es einen seitlichen Abstand von ¼Fensterlänge bzw. -höhe haben. Dies wird damit erreicht, dass das Fenster erst am Punkt w/4, h/4 angefangen wird zu zeichnen.

Java-Statements

Variablen:

```
static int              a;          //ganze Zahl
static double           b;          //reelle Zahl
static int[]            c;          //Zahlenpaket von ganzen Zahlen
static double[]         d;          //Zahlenpaket von reellen Zahlen
static String           f;          //Text
static JFrame           frame;      //Fenster
static JButton          b1;         //Knopf
```

Initialisierung der Variablen:

```
a=1;
b=2.21;
c=new int [3];
c[0]=1;
c[1]=2;
c[2]=3;
d=new double [2];
d[0]=4.23;
d[1]=9.55;
f=" Pneumonoultramicroscopicsilicovolcanoconiosis - Guinness World Record";
```

for-Schlaufe und if-Anweisungen:

```
for (int i=1;i<11;i++) {
      if (i!=10) System.out.print(i+", ");
      else System.out.print(i);
}
```

```
/*
 * Die Zahl i beginnt bei 1
 * Danach wird das ausgeführt, was in der FOR-Schlaufe
 * steht. Am Ende wird i jeweils um 1 erhöht.
 * Dies läuft so lange weiter, wie i kleiner als 11 ist,
 * also, bis i=10 ist.
 *
 * Die IF-Anweisung sagt, dass wenn i!=10 (!= heisst Ungleich)
 * ist, soll es immer ein Komma nach dem Ausgeschriebenen geben.
 * ELSE tritt dann in Kraft, wenn i!=10 nicht erfüllt ist (also i=10).
 *
 * System.out.print ist nichts anderes als ein Schreib-Befehl
 */
```

Mathematische Befehle

```
StrictMath.PI; //Zahl ∏;
StrictMath.cos(x); //cos(x)
StrictMath.sin(x); //sin(x)
StrictMath.sqrt(x); //√(x)
StrictMath.pow(x, y); //x^y
StrictMath.hypot(x, y); //√(x²/y²)
```

Das Programm

Die Berechnung der Anfangspunkte/-Geschwindigkeiten

Programm

```
public class Anfangspunkt {
	public static void main(String[] args) {
		double rAE,aAE,€,alpha,r,a,b,e,x,y,vx,vy,v,mt;
		//rp=Abstand von der Sonne am Perihel
		rAE=aAE=€=alpha=r=a=b=e=x=y=vx=vy=x=v=mt=0;
		double G=6.6742867E-11,mSonne = 1.989E30;
		for (int i=1;i<9;i++) {
			if (i==1) {
				rAE=0.314;
				aAE=0.38709888;
				€=0.20563069;
				alpha=49.49;
			}
			if (i==2) {
				rAE=0.723;
				aAE=0.72333193;
				€=0.00677323;
				alpha=46.01;
			}
			if (i==3) {
				aAE=1;
				€=0.01671022;
			}
			if (i==4) {
				rAE=1.666;
				aAE=1.5236621;
				€=0.09341233;
				alpha=153.50;
			}
			if (i==5) {
				rAE=4.985;
				aAE=5.2033623;
				€=0.04839266;
				alpha=341.81;
			}
			if (i==6) {
				rAE=9.5;
				aAE=9.537069;
				€=0.0541506;
				alpha=181.24;
			}
			if (i==7) {
				rAE=20.096;
				aAE=19.191261;
				€=0.04716771;
				alpha=356.51;
			}
			if (i==8) {
				rAE=30.022;
				aAE=30.06896;
				€=0.00858587;
				alpha=326.25;
```

```
		}
		//Umrechnung auf Meter
		r = rAE*1.4959789E11;
		a = aAE*1.4959789E11;
		e = a*€;
		b = StrictMath.sqrt(a*a-e*e);

		//Spezialfall Erde
		if (i==3) {
			x=-e;
			y=b;
			r=StrictMath.hypot(x, y);
		}
		//Spezialfall Uranus (Uranus hat den anderen Brennpunkt)
		if (i==7) {
			x=3.000538332E12;
			y=-1.829951399E11;
			/*
			 * Schnittpunkt von y=b*Wurzel(1-(x+e)^2/a^2)
			 * mit y=TAN(356.51°)*x:
			 */
			r=StrictMath.hypot(x, y);

		}

		//Berechnung x, y
		if (i!=3&&i!=7) {
			x = StrictMath.cos(alpha*StrictMath.PI/180.0)*r;
			y = StrictMath.sin(alpha*StrictMath.PI/180.0)*r;
		}

		//Vis-Viva-Gleichung v=Wurzel(GM(2/r-1/a)
		v=StrictMath.sqrt(G*mSonne*(2/r-1/a));

		//Berechnung vx/vy: Tangente an Ellipse am Punkt (x/y),
Steigung mt=vy/vx
		if (i!=7) {
			if (y>0) mt = -b*(x+e)/(StrictMath.sqrt(-x*x-
2*e*x+a*a-e*e)*a);
			if (y<=0)mt = +b*(x+e)/(StrictMath.sqrt(-x*x-
2*e*x+a*a-e*e)*a);
		}
		if (i==7) mt = b*(x-e)/(StrictMath.sqrt(-x*x+2*e*x+a*a-
e*e)*a);
		vx = v/StrictMath.hypot(mt, 1);
		vy=StrictMath.sqrt(v*v-vx*vx);
		if (i!=7) {
			if (y>0) vx=-vx;
			if (x<-e) vy=-vy;
		}
		if (i==7) {
			if (y>0) vx=-vx;
			if (x<e) vy=-vy;
		}

		//Ausgabe
		if (x>=0) System.out.println("\n x"+i+"= "+x);
		if (x<0) System.out.println("\n x"+i+"="+x);
		if (y>=0) System.out.println(" y"+i+"= "+y);
		if (y<0) System.out.println(" y"+i+"="+y);
```

```
                System.out.println("|v|= "+v);
                if (vx>=0) System.out.println("vx"+i+"= "+vx);
                if (vx<0) System.out.println("vx"+i+"="+vx);
                if (vy>=0) System.out.println("vy"+i+"= "+vy);
                if (vy<0) System.out.println("vy"+i+"="+vy);

            }
        }
}
```

Ausgabe

```
 x1= 3.0513235812278755E10
 y1= 3.5713785170789116E10
|v|= 57963.36038281667
vx1=-39912.08323917564
vy1= 42033.04364874611

 x2= 7.512016497661359E10
 y2= 7.781638302930272E10
|v|= 35041.893219567544
vx2=-24991.239339296106
vy2= 24563.636471369475

 x3=-2.4998136534358E9
 y3= 1.4957700232372018E11
|v|= 29789.079717521465
vx3=-29789.079717521465
vy3= 0.0

 x4=-2.230445667788105E11
 y4= 1.1120591877216039E11
|v|= 21974.70562188462
vx4=-10261.125627556183
vy4=-19431.854981550197

 x5= 7.084780069477961E11
 y5=-2.3279870504938177E11
|v|= 13619.187502125586
vx5= 3884.5851182367055
vy5= 13053.438883191957

 x6=-1.4208471420383193E12
 y6=-3.07548931934698E10
|v|= 9683.618314765268
vx6= 3261.538931589598
vy6=-9117.830194941267

 x7= 3.000538332E12
 y7=-1.829951399E11
|v|= 6487.050567174022
vx7= 414.40329664518174
vy7= 6473.800658716826

 x8= 3.734319482178555E12
 y8=-2.4951925051575684E12
|v|= 5440.967603689945
vx8= 2962.716388518755
vy8= 4563.599463757297
```


Das Hauptprogramm

Programm

```
import javax.swing.AbstractButton;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JFrame;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class MaturaArbeit   extends JPanel
                              implements ActionListener {
      //Auswahl
      static JButton b1,b2,b3,b4,b5,b6;
      static JFrame auswahlfenster,frame;
      //Anfangswerte:
      static int Aufgabe=0;
      static int[] w,h;
      static double x,y,wd,hd,m,n;
      //Sterne:
      static int u=0;
      static int[] sternx=new int [6],sterny=new int [6];
      //Datum
      static int day=25,month=3,year=2010;
      static String Jahr,Monat;
      //Reale Daten
      static double mSonne,mErde;
      //Rechnungsweg
      static int t;
      static double[] xt,yt,vx,vy,ax,ay;
      static double Mondx,Mondy,Mondvx,Mondvy,Mondax,Monday;
      //Konstanten
      static double G=6.6742867E-11;
      static double deltat=24*3600;
      //Zeichnen Planeten
      static int[] xdraw, ydraw;
      static double[] faktor;
      //Zeichnen Mond
      static int xMonddraw,yMonddraw;
      //Geschwindigkeitsreglung & Stop-Knopf-Regelung
      static int sleepN,stopsleepN,stopcounter=1;
      //PlanetenEllipsen zeichnen
      static int[]
merkurx,venusx,erdex,marsx,jupiterx,saturnx,uranusx,neptunx;
      static int[]
merkury,venusy,erdey,marsy,jupitery,saturny,uranusy,neptuny;
      static int[] ko,T;
      static double[] aAE,areal;
      //Fenstervergrösserung/-verkleinerung
      static boolean janein;
      //Anfangswerte
      static double[] Axt,Ayt,Avx,Avy;

      public static void main (String [] args) {
            //Geschwindigkeiten, Orte & Beschleunigungen
            vx = new double [8];
```

```
                vy = new double [8];
                ax = new double [8];
                ay = new double [8];
                xt = new double [8];
                yt = new double [8];
                xdraw = new int [8];
                ydraw = new int [8];

                //Höhe & Breite des Fensters
                w = new int [2];
                h = new int [2];

                //Verkleinerungsfaktor
                faktor = new double [2];

                //Berechnung der Umlaufzeiten (=Anzahl Punkte)
                areal = new double [8];
                aAE     = new double[8];
                T       = new int [8];
                areal[2]=1.4959789E11;
                T[2]=366;
                aAE[0]=0.38709888;
                aAE[1]=0.72333193;
                aAE[2]=1.0;
                aAE[3]=1.5236621;
                aAE[4]=5.2033623;
                aAE[5]=9.537069;
                aAE[6]=19.191261;
                aAE[7]=30.06896;
                for (int o=0;o<8;o++){
                      areal[o] = aAE[o]*areal[2];
                      T[o] = (int)
Math.round(StrictMath.sqrt(aAE[o]*aAE[o]*aAE[o]*(double)T[2]*(double)T[2]))
;
                }

//Werte (aus Wikipedia)
//
//Erde [2]:             Merkur[0]:        Venus[1]:         Mars[3]:
//a = 149597890km       a = 57909175km    a = 108208930km   a = 227936640km
//          1           0.38709888        0.72333193        1.5236621
<- relativ zur Erde
//€ = 0.01671022        € = 0.20563069    € = 0.00677323    € = 0.09341233
//t = 1j                t = 0.2408467j    t = 0.61519726j   t = 1.8808476j

//Jupiter[4]:           Saturn[5]:              Uranus[6]:
      Neptun[7]:
//a = 778.412.020km     a = 1.426.725.400km     a = 2.870.972.200km
      a = 4.498.252.900km
//5.2033623             9.5370690               19.191261
      30.068960         <- relativ zur Erde
//          0.2711319   0.4969485               1
      1.5668048         <- relativ zum Uranus
//€ = 0.04839266        € = 0.05415060          € = 0.04716771
      € = 0.00858587
//t = 11.862615j        t = 29.447498j          t = 84.016846j
      t = 164.79132j
//t =  0.402839         t = 1                   t =  2.853106
      t =   5.59611     <- relativ zum Saturn
```

```
            //Massen der Zentralkörper
            mSonne = 1.989E30;
            mErde= 5.9737E24;

            //Anfangswerte für 25. März 2010
            Axt=new double [8];
            Ayt=new double [8];
            Avx=new double [8];
            Avy=new double [8];
            Axt[0]=3.0513235812278755E10;       Ayt[0]=-
3.57137851707891l6E10;
            Avx[0]=-39912.08323917564;          Avy[0]=-42033.04364874611;
            Axt[1]=7.512016497661359E10;  Ayt[1]=-7.781638302930272E10;
            Avx[1]=24991.239339296106;          Avy[1]=24563.636471369475;
            /*Anfangswert Erde*/
            Axt[2]=-2.4998136534358E9;          Ayt[2]=-
1.49577002323720l8E11;
            Avx[2]=-29789.079717521465;   Avy[2]=0.0;
            for (int i=1;i<5;i++) {
                  ax[2]=-
G*mSonne*xt[2]/StrictMath.pow(StrictMath.hypot(Axt[2], Ayt[2]), 3);
                  ay[2]=-
G*mSonne*yt[2]/StrictMath.pow(StrictMath.hypot(Axt[2], Ayt[2]), 3);
                  Avx[2]=Avx[2]+ax[2]*deltat;
                  Avy[2]=Avy[2]+ay[2]*deltat;
                  Axt[2]=Axt[2]+Avx[2]*deltat;
                  Ayt[2]=Ayt[2]+Avy[2]*deltat;
            }
            Axt[3]=-2.230445667788105E11;       Ayt[3]=-
1.1120591877216039E11;
            Avx[3]=-10261.125627556183;   Avy[3]=19431.854981550197;
            Axt[4]=7.084780069477961E11;  Ayt[4]=2.3279870504938177E11;
            Avx[4]=3884.5851182367055;          Avy[4]=-13053.438883191957;
            Axt[5]=-1.4208471420383193E12;      Ayt[5]=3.07548931934698E10;
            Avx[5]=3261.538931589598;           Avy[5]=9117.830194941267;
            Axt[6]=3.000538332E12;              Ayt[6]=1.829951399E11;
            Avx[6]=414.4032923;                 Avy[6]=-6473.800658;
            Axt[7]=3.734319482178555E12;  Ayt[7]=2.4951925051575684E12;
            Avx[7]=2962.716388518755;           Avy[7]=-4563.599463757297;
            //Mondanfangswerte
            Mondx=363300E3;
            Mondy=0;
            Mondvx=0;
            Mondvy=-1076;

            if (Aufgabe==0) {
                  javax.swing.SwingUtilities.invokeLater(new Runnable()
{public void run() {Auswahlfenster();}});
            }
            try {
                  Thread.sleep(2000);
                  b1.setVisible(true);
                  b2.setVisible(true);
            }
            catch (Exception ex){
                  System.out.print("ERROR in main!");
            }

            ko = new int [8];
            for (int i=0;i<8;i++) ko[i]=0;
```

```
while (true) {
	if (Aufgabe==1) {
		auswahlfenster.setVisible(false);
		w[0]=900;
		h[0]=600;
		faktor[0]=5E11/(h[0]-50);
		merkurx= new int [T[0]];
		merkury= new int [T[0]];
		venusx = new int [T[1]];
		venusy = new int [T[1]];
		erdex = new int [T[2]];
		erdey = new int [T[2]];
		marsx = new int [T[3]];
		marsy = new int [T[3]];
		for (int i=0;i<4;i++) {
			xt[i]=Axt[i];
			yt[i]=Ayt[i];
			vx[i]=Avx[i];
			vy[i]=Avy[i];
			for (int abc=0;abc<T[i];abc++) {
				ax[i]=-
G*mSonne*xt[i]/StrictMath.pow(StrictMath.hypot(xt[i], yt[i]), 3);
				ay[i]=-
G*mSonne*yt[i]/StrictMath.pow(StrictMath.hypot(xt[i], yt[i]), 3);
				if(abc==0&&i!=2) {
					vx[i]=vx[i]+ax[i]*deltat*0.5;
					vy[i]=vy[i]+ay[i]*deltat*0.5;
				}
				vx[i]=vx[i]+ax[i]*deltat;
				vy[i]=vy[i]+ay[i]*deltat;
				xt[i]=xt[i]+vx[i]*deltat;
				yt[i]=yt[i]+vy[i]*deltat;
				xdraw[i]=
(int)Math.round(xt[i]/(faktor[0]));
				ydraw[i]=
(int)Math.round(yt[i]/(faktor[0]));
				if (i==0) {
					merkurx[abc]=w[0]/2+xdraw[i];
					merkury[abc]=h[0]/2+ydraw[i];
				}
				if (i==1) {
					venusx[abc]=w[0]/2+xdraw[i];
					venusy[abc]=h[0]/2+ydraw[i];
				}
				if (i==2) {
					erdex[abc]=w[0]/2+xdraw[i];
					erdey[abc]=h[0]/2+ydraw[i];
				}
				if (i==3) {
					marsx[abc]=w[0]/2+xdraw[i];
					marsy[abc]=h[0]/2+ydraw[i];
				}
			}
		}
		for (int i=0;i<4;i++) {
			xt[i]=Axt[i];
			yt[i]=Ayt[i];
			vx[i]=Avx[i];
			vy[i]=Avy[i];
		}
```

```
                        new MaturaArbeit().run();
                  }
                  if (Aufgabe==2) {
                        auswahlfenster.setVisible(false);
                        w[0]=792;
                        h[0]=401;
                        faktor[0]=5E11/(h[0]-50);
                        erdex = new int [T[2]];
                        erdey = new int [T[2]];
                        int i=2;
                        xt[i]=Axt[i];
                        yt[i]=Ayt[i];
                        vx[i]=Avx[i];
                        vy[i]=Avy[i];
                        for (int abc=0;abc<T[i];abc++) {
                              ax[i]=-
G*mSonne*xt[i]/StrictMath.pow(StrictMath.hypot(xt[i], yt[i]), 3);
                              ay[i]=-
G*mSonne*yt[i]/StrictMath.pow(StrictMath.hypot(xt[i], yt[i]), 3);
                              vx[i]=vx[i]+ax[i]*deltat;
                              vy[i]=vy[i]+ay[i]*deltat;
                              xt[i]=xt[i]+vx[i]*deltat;
                              yt[i]=yt[i]+vy[i]*deltat;
                              xdraw[i]= (int)Math.round(xt[i]/(faktor[0]));
                              ydraw[i]= (int)Math.round(yt[i]/(faktor[0]));
                              erdex[abc]=w[0]/2+xdraw[i];
                              erdey[abc]=h[0]/2+ydraw[i];
                        }
                        new MaturaArbeit().run2();
                  }
                  if (Aufgabe==3) {
                        auswahlfenster.setVisible(false);
                        w[0]=962;
                        h[0]=961;
                        faktor[0]=1E12/(h[0]-50);
                        jupiterx = new int [T[4]];
                        jupitery = new int [T[4]];
                        saturnx = new int [T[5]];
                        saturny = new int [T[5]];
                        uranusx = new int [T[6]];
                        uranusy = new int [T[6]];
                        neptunx = new int [T[7]];
                        neptuny = new int [T[7]];
                        for (int i=4;i<8;i++) {
                              xt[i]=Axt[i];
                              yt[i]=Ayt[i];
                              vx[i]=Avx[i];
                              vy[i]=Avy[i];
                              for (int abc=0;abc<T[i];abc++) {
                                    ax[i]=-
G*mSonne*xt[i]/StrictMath.pow(StrictMath.hypot(xt[i], yt[i]), 3);
                                    ay[i]=-
G*mSonne*yt[i]/StrictMath.pow(StrictMath.hypot(xt[i], yt[i]), 3);
                                    if(abc==0) {
                                          vx[i]=vx[i]+ax[i]*deltat*0.5;
                                          vy[i]=vy[i]+ay[i]*deltat*0.5;
                                    }
                                    vx[i]=vx[i]+ax[i]*deltat;
                                    vy[i]=vy[i]+ay[i]*deltat;
                                    xt[i]=xt[i]+vx[i]*deltat;
```

```
                                        yt[i]=yt[i]+vy[i]*deltat;
                                        xdraw[i]=
(int)Math.round(xt[i]/(faktor[0]*10));
                                        ydraw[i]=
(int)Math.round(yt[i]/(faktor[0]*10));
                                        if (i==4) {
                                              jupiterx[abc]=w[0]/2+xdraw[i];
                                              jupitery[abc]=h[0]/2+ydraw[i];
                                        }
                                        if (i==5) {
                                              saturnx[abc]=w[0]/2+xdraw[i];
                                              saturny[abc]=h[0]/2+ydraw[i];
                                        }
                                        if (i==6) {
                                              uranusx[abc]=w[0]/2+xdraw[i];
                                              uranusy[abc]=h[0]/2+ydraw[i];
                                        }
                                        if (i==7) {
                                              neptunx[abc]=w[0]/2+xdraw[i];
                                              neptuny[abc]=h[0]/2+ydraw[i];
                                        }
                                  }
                            }
                            for (int i=4;i<8;i++) {
                                  xt[i]=Axt[i];
                                  yt[i]=Ayt[i];
                                  vx[i]=Avx[i];
                                  vy[i]=Avy[i];
                            }
                            new MaturaArbeit().run3();
                      }
                }
          }

          public MaturaArbeit() {
                //Knopf 1
                b1 = new JButton("Innere Planeten");
                b1.setVerticalTextPosition(AbstractButton.CENTER);
                b1.setHorizontalTextPosition(AbstractButton.CENTER);
                b1.setActionCommand("Innen");
                b1.addActionListener(this);
                add(b1);

                //Knopf 2
                b2 = new JButton("Planete Erde");
                b2.setVerticalTextPosition(AbstractButton.CENTER);
                b2.setHorizontalTextPosition(AbstractButton.CENTER);
                b2.setActionCommand("Erde");
                b2.addActionListener(this);
                add(b2);

                //Knopf 3
                b3 = new JButton("Äussere Planeten");
                b3.setVerticalTextPosition(AbstractButton.CENTER);
                b3.setHorizontalTextPosition(AbstractButton.CENTER);
                b3.setActionCommand("Aussen");
                b3.addActionListener(this);
                add(b3);
```

```
        //Text beim Darüberfahren mit der Maus
        b1.setToolTipText("Merkur - Venus - Erde - Mars");
        b2.setToolTipText("Erde");
        b3.setToolTipText("Jupiter - Saturn - Uranus - Neptun");

        //Knopf Schneller
        b4 = new JButton("Schneller");
        b4.setVerticalTextPosition(AbstractButton.CENTER);
        b4.setHorizontalTextPosition(AbstractButton.CENTER);
        b4.setActionCommand("faster");
        b4.addActionListener(this);
        b4.setVisible(false);
        add(b4);

        //Knopf Längsämer
        b5 = new JButton("Längsämer");
        b5.setVerticalTextPosition(AbstractButton.CENTER);
        b5.setHorizontalTextPosition(AbstractButton.CENTER);
        b5.setActionCommand("slower");
        b5.addActionListener(this);
        b5.setVisible(false);
        add(b5);

        //Knopf Stop
        b6 = new JButton("STOP");
        b6.setVerticalTextPosition(AbstractButton.CENTER);
        b6.setHorizontalTextPosition(AbstractButton.CENTER);
        b6.setActionCommand("sop");
        b6.addActionListener(this);
        b6.setVisible(false);
        add(b6);

        if (Aufgabe>0) {
              b1.setVisible(false);
              b2.setVisible(false);
              b3.setVisible(false);
              b4.setVisible(true);
              b5.setVisible(true);
              b6.setVisible(true);
        }
}

public void actionPerformed(ActionEvent e) {
        if ("Innen".equals(e.getActionCommand())) {
        Aufgabe=1;
   }
        if ("Erde".equals(e.getActionCommand())) {
        Aufgabe=2;
   }
        if ("Aussen".equals(e.getActionCommand())) {
        Aufgabe=3;
   }
        if ("faster".equals(e.getActionCommand())) {
              if (sleepN>19&&sleepN<101)sleepN=sleepN-10;
              if (sleepN>100&&sleepN<201) sleepN=sleepN-20;
              if (sleepN>200) sleepN=sleepN-50;
        }
        if ("slower".equals(e.getActionCommand())) {
              if (sleepN<100)sleepN=sleepN+10;
              if (sleepN>99&&sleepN<200) sleepN=sleepN+20;
```

```
                if (sleepN>199) sleepN=sleepN+50;
            }
            if ("sop".equals(e.getActionCommand())) {
                if (stopcounter==1){
                    stopsleepN=sleepN;
                    sleepN=1000;
                    b6.setText("LOS");
                    stopcounter--;
                }
                else {
                    b6.setText("STOP");
                    if (stopsleepN==0) stopsleepN=sleepN;
                    sleepN=stopsleepN;
                    stopcounter++;
                }
            }
        }

        public static void Auswahlfenster() {
            //Neues Fenster
            auswahlfenster = new JFrame("Auswahlfenster");
            auswahlfenster.setVisible(true);
            auswahlfenster.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

          //Inhalt des Fensters
          MaturaArbeit Auswahlfenster = new MaturaArbeit();
          Auswahlfenster.setOpaque(true);
          auswahlfenster.setContentPane(Auswahlfenster);
          auswahlfenster.pack();
        }

        public void run() {
            Aufgabe=1;
            sleepN=50;
            frame = new JFrame();
            frame.setSize(916,638);
            frame.setTitle("Innere Planeten");
            frame.getContentPane().add(this);
            frame.setVisible(true);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            //Nachfolgende Werte
            try{
                for (t=734152;true;t++) {
                    if (stopcounter==1) {
                        for (int ii=0;ii<4;ii++) {
                            ax[ii]=-
G*mSonne*xt[ii]/StrictMath.pow(StrictMath.hypot(xt[ii], yt[ii]), 3);
                            ay[ii]=-
G*mSonne*yt[ii]/StrictMath.pow(StrictMath.hypot(xt[ii], yt[ii]), 3);
                            if(t==734152&&ii!=2) {
                                vx[ii]=vx[ii]+ax[ii]*deltat*0.5;
                                vy[ii]=vy[ii]+ay[ii]*deltat*0.5;
                            }
                            vx[ii]=vx[ii]+ax[ii]*deltat;
                            vy[ii]=vy[ii]+ay[ii]*deltat;
                            xt[ii]=xt[ii]+vx[ii]*deltat;
                            yt[ii]=yt[ii]+vy[ii]*deltat;
                            /*xt = xt + v(t+dt/2)*dt
                              v(t+dt/2) = v(t-dt/2)+a(t)*dt
                              a(t)=G*M*x(t)/r(t)^3
```

```
                                        */
                                    }
                                    //Mond
                                    Mondax=-
G*mErde/StrictMath.pow(StrictMath.hypot(Mondx, Mondy), 3)*Mondx;
                                    Monday=-
G*mErde/StrictMath.pow(StrictMath.hypot(Mondx, Mondy), 3)*Mondy;
                                    Mondvx=Mondvx+Mondax*deltat;
                                    Mondvy=Mondvy+Monday*deltat;
                                    Mondx=Mondx+Mondvx*deltat;
                                    Mondy=Mondy+Mondvy*deltat;
                                    //Datum
                                    day++;
                                    if (year%4!=0) if (day==29&&month==2) {
                                          day=1;
                                          month++;
                                    }
                                    if (year%4==0)
                                          if (day==30&&month==2) {
                                                day=1;month++;
                                          }
                                    if (day==31) {
                                          if (month<8&&month%2==0&&month!=2){
                                                day=1;
                                                month++;
                                          }
                                          if (month>7&&month%2!=0){
                                                day=1;
                                                month++;
                                          }
                                    }
                                    if (day==32) {
                                          if (month>7&&month%2==0){
                                                day=1;
                                                month++;
                                          }
                                          if (month<8&&month%2!=0){
                                                day=1;
                                                month++;
                                          }
                                    }
                                    if (month==13) {
                                          month=1;
                                          year++;
                                    }
                                    repaint();
                              }
                              Thread.sleep(sleepN);
                              if (stopcounter==0) t--;
                        }
                  }
                  catch (Exception pi) {
                        System.out.println("Error in run()");
                  }
            }

            public void run2() {
                  Aufgabe=2;
                  sleepN=50;
                  frame = new JFrame();
```

```
		frame.setSize(808,439);
		frame.setTitle("Erde & Mond");
		frame.getContentPane().add(this);
		frame.setVisible(true);
		frame.setResizable(true);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		try {
			int i=2;
			//Nachfolgende Werte
			for (t=734152;true;t++) {
				if (stopcounter==1) {
					ax[i]=-
G*mSonne/StrictMath.pow(StrictMath.hypot(xt[i], yt[i]), 3)*xt[i];
					ay[i]=-
G*mSonne/StrictMath.pow(StrictMath.hypot(xt[i], yt[i]), 3)*yt[i];
					vx[i]=vx[i]+ax[i]*deltat;
					vy[i]=vy[i]+ay[i]*deltat;
					xt[i]=xt[i]+vx[i]*deltat;
					yt[i]=yt[i]+vy[i]*deltat;
					/*xt = xt + v(t+dt/2)*dt
					v(t+dt/2) = v(t-dt/2)+a(t)*dt
					a(t)=G*M*x(t)/r(t)^3
					*/
					//Mond
					Mondax=-
G*mErde/StrictMath.pow(StrictMath.hypot(Mondx, Mondy), 3)*Mondx;
					Monday=-
G*mErde/StrictMath.pow(StrictMath.hypot(Mondx, Mondy), 3)*Mondy;
					Mondvx=Mondvx+Mondax*deltat;
					Mondvy=Mondvy+Monday*deltat;
					Mondx=Mondx+Mondvx*deltat;
					Mondy=Mondy+Mondvy*deltat;
					//Datum
					day++;
					if (year%4!=0) if (day==29&&month==2) {
						day=1;
						month++;
					}
					if (year%4==0)
						if (day==30&&month==2) {
							day=1;month++;
						}
					if (day==31) {
						if (month<8&&month%2==0&&month!=2){
							day=1;
							month++;
						}
						if (month>7&&month%2!=0){
							day=1;
							month++;
						}
					}
					if (day==32) {
						if (month>7&&month%2==0){
							day=1;
							month++;
						}
						if (month<8&&month%2!=0){
							day=1;
							month++;
```

```
                                        }
                                  }
                                  if (month==13) {
                                        month=1;
                                        year++;
                                  }
                                  repaint();
                            }
                            Thread.sleep(sleepN);
                            if (stopcounter==0) t--;
                      }
                }
                catch (Exception pi) {
                      System.out.println("Error in run3()");
                }
          }

          public void run3() {
                Aufgabe=3;
                sleepN=10;
                frame = new JFrame();
                frame.setSize(968,989);
                frame.setTitle("Äussere Planeten");
                frame.getContentPane().add(this);
                frame.setVisible(true);
                frame.setResizable(true);
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                //Nachfolgende Werte
                try{
                      for (t=734152;true;t++) {
                            if (stopcounter==1) {
                                  for (int ii=4;ii<8;ii++) {
                                        ax[ii]=-
G*mSonne*xt[ii]/StrictMath.pow(StrictMath.hypot(xt[ii], yt[ii]), 3);
                                        ay[ii]=-
G*mSonne*yt[ii]/StrictMath.pow(StrictMath.hypot(xt[ii], yt[ii]), 3);
                                        if(t==734152) {
                                              vx[ii]=vx[ii]+ax[ii]*deltat*0.5;
                                              vy[ii]=vy[ii]+ay[ii]*deltat*0.5;
                                        }
                                        vx[ii]=vx[ii]+ax[ii]*deltat;
                                        vy[ii]=vy[ii]+ay[ii]*deltat;
                                        xt[ii]=xt[ii]+vx[ii]*deltat;
                                        yt[ii]=yt[ii]+vy[ii]*deltat;
                                        /*xt = xt + v(t+dt/2)*dt
                                          v(t+dt/2) = v(t-dt/2)+a(t)*dt
                                          a(t)=G*M*x(t)/r(t)^3
                                         */
                                        /*{
                                              int jott=5;
                                              xdraw[jott]=
(int)Math.round(xt[jott]/(faktor*10));
                                              ydraw[jott]=
(int)Math.round(yt[jott]/(faktor*10));
                                              if (xdraw[jott]>75&&ydraw[jott]<-
45)System.out.println("x="+xdraw[jott]+" y="+ydraw[jott]);
                                              if
(xdraw[jott]==82&&ydraw[jott]==-52)System.out.println(t);
                                        }*/
                                  }
```


```
                        //Datum
                        day++;
                        if (year%4!=0) if (day==29&&month==2) {
                              day=1;
                              month++;
                        }
                        if (year%4==0) {
                              if (day==30&&month==2) {
                                    day=1;month++;
                              }
                        }
                        if (day==31) {
                              if (month<8&&month%2==0&&month!=2){
                                    day=1;
                                    month++;
                              }
                              if (month>7&&month%2!=0){
                                    day=1;
                                    month++;
                              }
                        }
                        if (day==32) {
                              if (month>7&&month%2==0){
                                    day=1;
                                    month++;
                              }
                              if (month<8&&month%2!=0){
                                    day=1;
                                    month++;
                              }
                        }
                        if (month==13) {
                              month=1;
                              year++;
                        }
                        repaint();
                        /*xt = xt + v(t+dt/2)*dt
                        v(t+dt/2) = v(t-dt/2)+a(t)*dt
                        a(t)=G*M*x(t)/r(t)^3
                        */
                  }
                  Thread.sleep(sleepN);
                  if (stopcounter==0) t--;
            }
      }
      catch (Exception pi) {
            System.out.println("Error in run()");
      }
}

public void paintComponent(Graphics g) {
      super.paintComponent(g);
      try {
            if (Aufgabe>0) {
                  janein=true;
                  g.setColor(Color.green);
                  g.drawString("Geschwindigkeit =
"+(int)(1.0/(0.001*(double)sleepN))+" Tage/Sekunde", 10, 20);
                  w[1]=w[0];
                  h[1]=h[0];
```

```
                        w[0] = getWidth();
                        h[0] = getHeight();
                        wd = (double) w[0];
                        hd = (double) h[0];
                        faktor[1]=faktor[0];
                        m = StrictMath.min(wd, hd);
                        if (Aufgabe<3)faktor[0]=5E11/(m-50.0);
                        if (Aufgabe==3) faktor[0]=1E12/(m-50.0);
                        n = (int) StrictMath.round(StrictMath.max(wd, hd));
                        setBackground(Color.black);
                        g.setColor(Color.white);

                        if (h[0]!=h[1]) janein=false;
                        if (w[0]!=w[1]) janein=false;

                        if (janein==false) {
                              stopcounter=0;
                              Thread.sleep(1000);
                              if (Aufgabe==1) {
                                    for (int i=0;i<4;i++) {
                                          for (int abc=0;abc<T[i];abc++) {
                                                if (i==0) {
                                                      merkurx[abc]=(int)
Math.round((merkurx[abc]-w[1]/2)*faktor[1]/faktor[0]+w[0]/2);
                                                      merkury[abc]=(int)
Math.round((merkury[abc]-h[1]/2)*faktor[1]/faktor[0]+h[0]/2);
                                                }
                                                if (i==1) {
                                                      venusx[abc]=(int)
Math.round((venusx[abc]-w[1]/2)*faktor[1]/faktor[0]+w[0]/2);
                                                      venusy[abc]=(int)
Math.round((venusy[abc]-h[1]/2)*faktor[1]/faktor[0]+h[0]/2);
                                                }
                                                if (i==2) {
                                                      erdex[abc]=(int)
Math.round((erdex[abc]-w[1]/2)*faktor[1]/faktor[0]+w[0]/2);
                                                      erdey[abc]=(int)
Math.round((erdey[abc]-h[1]/2)*faktor[1]/faktor[0]+h[0]/2);
                                                }
                                                if (i==3) {
                                                      marsx[abc]=(int)
Math.round((marsx[abc]-w[1]/2)*faktor[1]/faktor[0]+w[0]/2);
                                                      marsy[abc]=(int)
Math.round((marsy[abc]-h[1]/2)*faktor[1]/faktor[0]+h[0]/2);
                                                }
                                          }
                                    }
                              }
                              if (Aufgabe==2) {
                                    int i=2;
                                    for (int abc=0;abc<T[i];abc++) {
                                          erdex[abc]=(int)
Math.round((erdex[abc]-w[1]/2)*faktor[1]/faktor[0]+w[0]/2);
                                          erdey[abc]=(int)
Math.round((erdey[abc]-h[1]/2)*faktor[1]/faktor[0]+h[0]/2);
                                    }
                              }
                              if (Aufgabe==3) {
                                    for (int i=4;i<8;i++) {
                                          for (int abc=0;abc<T[i];abc++) {
```

```
                                                if (i==4) {
                                                      jupiterx[abc]=(int)
Math.round((jupiterx[abc]-w[1]/2)*faktor[1]/faktor[0]+w[0]/2);
                                                      jupitery[abc]=(int)
Math.round((jupitery[abc]-h[1]/2)*faktor[1]/faktor[0]+h[0]/2);
                                                }
                                                if (i==5) {
                                                      saturnx[abc]=(int)
Math.round((saturnx[abc]-w[1]/2)*faktor[1]/faktor[0]+w[0]/2);
                                                      saturny[abc]=(int)
Math.round((saturny[abc]-h[1]/2)*faktor[1]/faktor[0]+h[0]/2);
                                                }
                                                if (i==6) {
                                                      uranusx[abc]=(int)
Math.round((uranusx[abc]-w[1]/2)*faktor[1]/faktor[0]+w[0]/2);
                                                      uranusy[abc]=(int)
Math.round((uranusy[abc]-h[1]/2)*faktor[1]/faktor[0]+h[0]/2);
                                                }
                                                if (i==7) {
                                                      neptunx[abc]=(int)
Math.round((neptunx[abc]-w[1]/2)*faktor[1]/faktor[0]+w[0]/2);
                                                      neptuny[abc]=(int)
Math.round((neptuny[abc]-h[1]/2)*faktor[1]/faktor[0]+h[0]/2);
                                                }
                                          }
                                    }
                              }
                              stopcounter=1;
                        }
                        // Sterne zeichnen
                        for (u=0;u<12;u++) {
                              if (u<6) {
                                    if (t%4==0) {
                                          sternx[u] = (int)
StrictMath.round((w[0]-2)*StrictMath.random());
                                          sterny[u] = (int)
StrictMath.round((h[0]-4)*StrictMath.random());
                                    }
                                    g.drawLine(sternx[u], sterny[u],
sternx[u], sterny[u]+4);
                                    g.drawLine(sternx[u]-2, sterny[u]+2,
sternx[u]+2, sterny[u]+2);
                              }
                              else {
                                    int steernx=(int)
StrictMath.round((w[0]-2)*StrictMath.random());
                                    int steerny=(int)
StrictMath.round((h[0]-4)*StrictMath.random());
                                    g.drawLine(steernx, steerny, steernx,
steerny);
                              }
                        }

                        //Sonne
                        g.setColor(Color.yellow);
                        g.fillOval(w[0]/2-20,h[0]/2-20,40,40);

                        //Datum
                        if (month==1)     Monat="Januar";
                        if (month==2)     Monat="Februar";
```

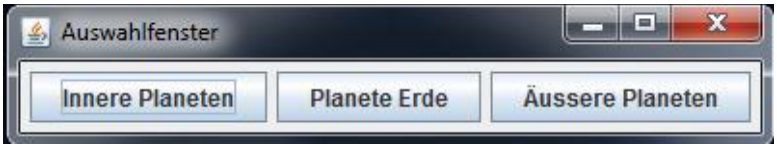
```
                        if (month==3)     Monat="März";
                        if (month==4)     Monat="April";
                        if (month==5)     Monat="Mai";
                        if (month==6)     Monat="Juni";
                        if (month==7)     Monat="Juli";
                        if (month==8)     Monat="August";
                        if (month==9)     Monat="September";
                        if (month==10)    Monat="Oktober";
                        if (month==11)    Monat="November";
                        if (month==12)    Monat="Dezember";
                        Jahr=""+year;

                        if (Aufgabe==1) {
                              //Faktormethode:
                              for (int i=0;i<4;i++) {
                                    xdraw[i]=
(int)Math.round(xt[i]/faktor[0]);
                                    ydraw[i]=
(int)Math.round(yt[i]/faktor[0]);
                              }

      xMonddraw=xdraw[2]+(int)Math.round(Mondx/(1E9/50));

      yMonddraw=ydraw[2]+(int)Math.round(Mondy/(1E9/50));
                              //Elipsen der Planeten Innen
                              g.setColor(new Color(50,50,50));
                              for (int i=0;i<4;i++) {
                                    if (i==0) {
                                          if (ko[i]==T[i]) ko[i]=0;
                                          merkurx[ko[i]]=w[0]/2+xdraw[i];
                                          merkury[ko[i]]=h[0]/2+ydraw[i];
                                    }
                                    if (i==1) {
                                          if (ko[i]==T[i]) ko[i]=0;
                                          venusx[ko[i]]=w[0]/2+xdraw[i];
                                          venusy[ko[i]]=h[0]/2+ydraw[i];
                                    }
                                    if (i==2) {
                                          if (ko[i]==T[i]) ko[i]=0;
                                          erdex[ko[i]]=w[0]/2+xdraw[i];
                                          erdey[ko[i]]=h[0]/2+ydraw[i];
                                    }
                                    if (i==3) {
                                          if (ko[i]==T[i]) ko[i]=0;
                                          marsx[ko[i]]=w[0]/2+xdraw[i];
                                          marsy[ko[i]]=h[0]/2+ydraw[i];
                                    }
                                    ko[i]++;
                              }
                              g.setColor(Color.DARK_GRAY);
                              g.drawPolygon(merkurx, merkury, T[0]);
                              g.setColor(new Color(100,100,0));
                              g.drawPolygon(venusx, venusy, T[1]);
                              g.setColor(new Color(40, 40, 80));
                              g.drawPolygon(erdex, erdey, T[2]);
                              g.setColor(new Color(100, 0, 0));
                              g.drawPolygon(marsx, marsy, T[3]);
                              //Merkur
                              g.setColor(Color.DARK_GRAY);
```

```
                                    g.fillOval(w[0]/2+xdraw[0]-5, h[0]/2+ydraw[0]-
5, 10, 10);
                                    //Venus
                                    g.setColor(Color.ORANGE);
                                    g.fillOval(w[0]/2+xdraw[1]-10,
h[0]/2+ydraw[1]-10, 20, 20);
                                    //Erde
                                    g.setColor(Color.CYAN);
                                    g.fillOval(w[0]/2+xdraw[2]-10,
h[0]/2+ydraw[2]-10, 20, 20);
                                    //Mond
                                    g.setColor(Color.white);
                                    g.fillOval(w[0]/2+xMonddraw-3,
h[0]/2+yMonddraw-3, 6, 6);
                                    //Mars
                                    g.setColor(Color.RED);
                                    g.fillOval(w[0]/2+xdraw[3]-7, h[0]/2+ydraw[3]-
7, 14, 14);
                              }

                              if (Aufgabe==2) {
                                    xdraw[2]= (int)Math.round(xt[2]/faktor[0]);
                                    ydraw[2]= (int)Math.round(yt[2]/faktor[0]);

      xMonddraw=xdraw[2]+(int)Math.round(Mondx/(faktor[0]/50));

      yMonddraw=ydraw[2]+(int)Math.round(Mondy/(faktor[0]/50));
                                    //Elipse des Planeten Erde
                                    if (ko[2]==T[2]) ko[2]=0;
                                    erdex[ko[2]]=w[0]/2+xdraw[2];
                                    erdey[ko[2]]=h[0]/2+ydraw[2];
                                    ko[2]++;
                                    g.setColor(new Color(40, 40, 80));
                                    g.drawPolygon(erdex, erdey, T[2]);
                                    //Erde
                                    g.setColor(Color.CYAN);
                                    g.fillOval(w[0]/2+xdraw[2]-10,
h[0]/2+ydraw[2]-10, 20, 20);
                                    //Mond
                                    g.setColor(Color.white);
                                    g.fillOval(w[0]/2+xMonddraw-3,
h[0]/2+yMonddraw-3, 6, 6);
                              }
                              if (Aufgabe==3) {
                                    //Faktormethode:
                                    for (int i=4;i<8;i++) {
                                          xdraw[i]=
(int)Math.round(xt[i]/(faktor[0]*10));
                                          ydraw[i]=
(int)Math.round(yt[i]/(faktor[0]*10));
                                    }
                                    //Elipsen der Planeten Aussen
                                    for (int i=4;i<8;i++) {
                                          if (i==4) {
                                                if (ko[i]==T[i]) ko[i]=0;
                                                jupiterx[ko[i]]=w[0]/2+xdraw[i];
                                                jupitery[ko[i]]=h[0]/2+ydraw[i];
                                          }
                                          if (i==5) {
                                                if (ko[i]==T[i]) ko[i]=0;
```

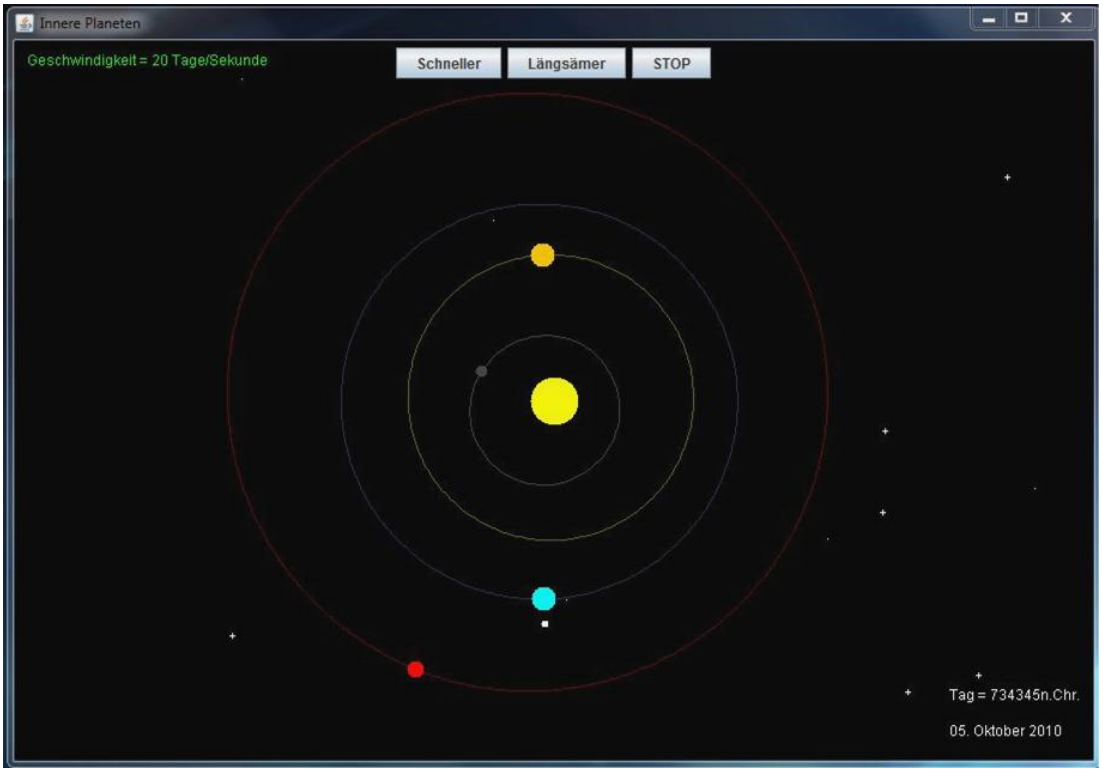
```
                                    saturnx[ko[i]]=w[0]/2+xdraw[i];
                                    saturny[ko[i]]=h[0]/2+ydraw[i];
                              }
                              if (i==6) {
                                    if (ko[i]==T[i]) ko[i]=0;
                                    uranusx[ko[i]]=w[0]/2+xdraw[i];
                                    uranusy[ko[i]]=h[0]/2+ydraw[i];
                              }
                              if (i==7) {
                                    if (ko[i]==T[i]) ko[i]=0;
                                    neptunx[ko[i]]=w[0]/2+xdraw[i];
                                    neptuny[ko[i]]=h[0]/2+ydraw[i];
                              }
                              ko[i]++;
                        }
                        g.setColor(new Color(120,70,0));
                        g.drawPolygon(jupiterx, jupitery, T[4]);
                        g.setColor(new Color(80,80,10));
                        g.drawPolygon(saturnx, saturny, T[5]);
                        g.setColor(new Color(0,70,80));
                        g.drawPolygon(uranusx, uranusy, T[6]);
                        g.setColor(new Color(0,20,80));
                        g.drawPolygon(neptunx, neptuny, T[7]);
                        //Jupiter
                        g.setColor(Color.ORANGE);
                        g.fillOval(w[0]/2+xdraw[4]-12,
h[0]/2+ydraw[4]-12, 24, 24);
                        //Saturn
                        g.setColor(Color.YELLOW);
                        g.fillOval(w[0]/2+xdraw[5]-8, h[0]/2+ydraw[5]-
8, 16, 16);
                        g.setColor(Color.WHITE);
                        g.drawOval(w[0]/2+xdraw[5]-10,
h[0]/2+ydraw[5]-10, 20, 20);
                        //Uranus
                        g.setColor(Color.CYAN);
                        g.fillOval(w[0]/2+xdraw[6]-6, h[0]/2+ydraw[6]-
6, 12, 12);
                        //Neptun
                        g.setColor(Color.BLUE);
                        g.fillOval(w[0]/2+xdraw[7]-6, h[0]/2+ydraw[7]-
6, 12, 12);
                  }
                  g.setColor(Color.white);
                  g.drawString("Tag = "+t+"n.Chr.", w[0]-120, h[0]-
50);
                  if (day<10) g.drawString("0"+day+". "+Monat+"
"+Jahr, w[0]-120, h[0]-20);
                  if (day>9) g.drawString(day+". "+Monat+" "+Jahr,
w[0]-120, h[0]-20);
            }
      }
      catch (Exception exc) {
            System.out.println("PAINT ERROR");
      }
  }
}
```

Ausgabe:

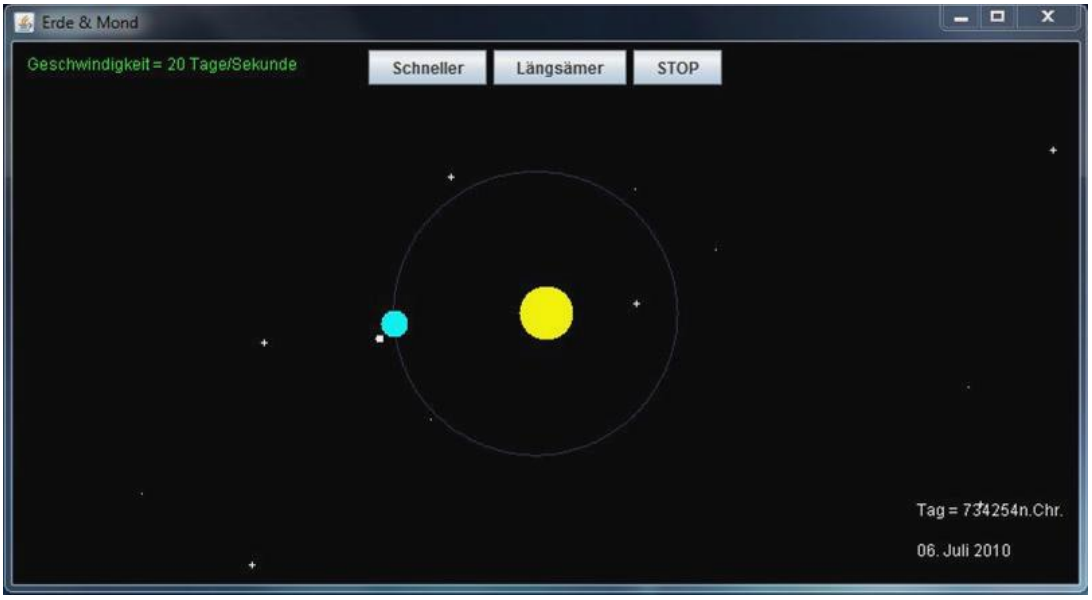
Auswahlfenster:



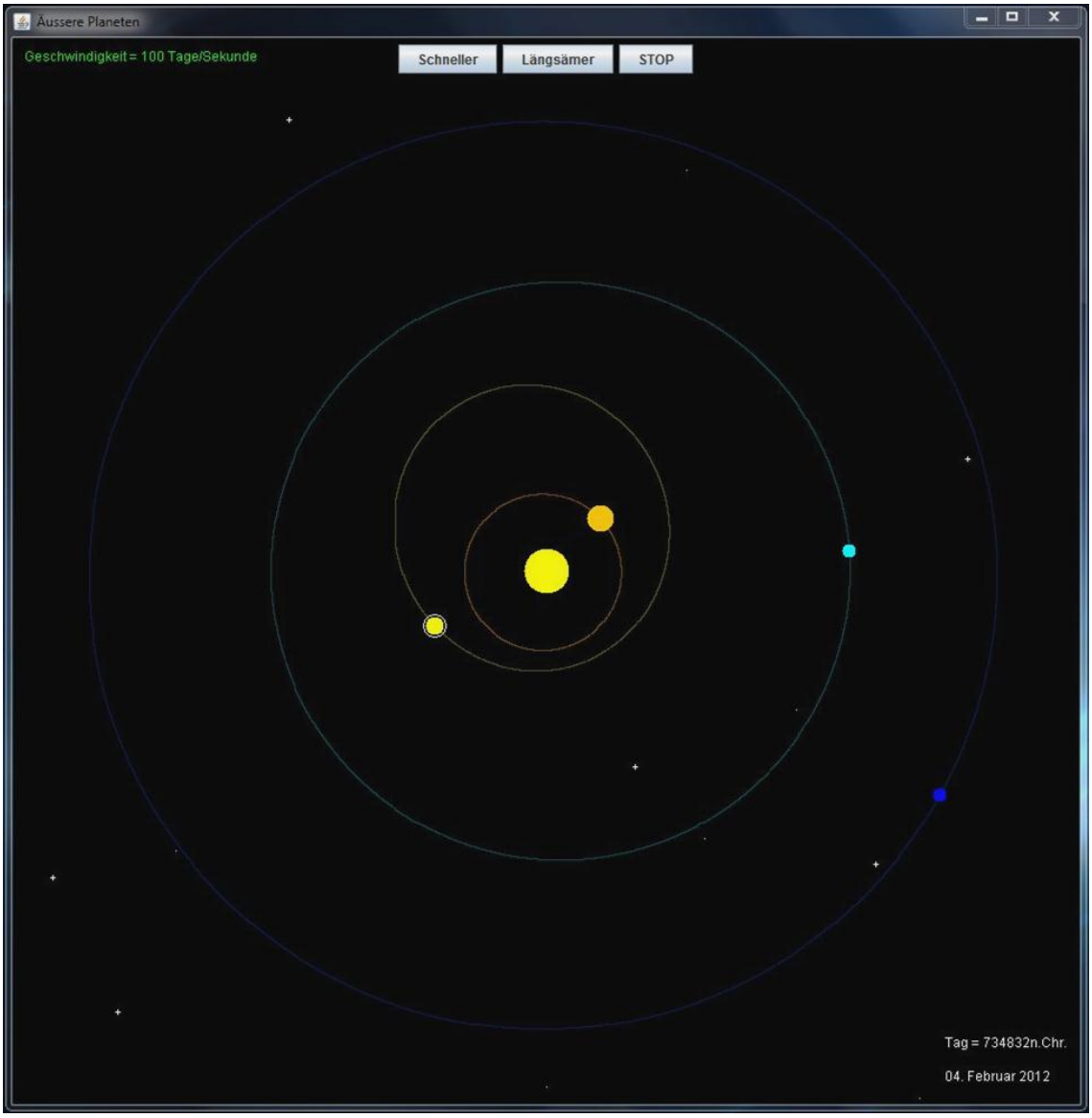
Innere Planeten



Erde allein



Äussere Planeten



Anhang

- CD mit folgendem Inhalt:
  - Titelblatt zur Matura-Arbeit in PDF-Version
  - Matura-Arbeit in PDF-Version
  - Titelblatt zum Arbeitsprotokoll in PDF-Version
  - Arbeitsprotokoll zur Matura-Arbeit in PDF-Version
  - Das Programm in .java-Datei und in .jar-Datei
  - Java™ SE Development Kit 7u1
- Arbeitsprotokoll zur Matura-Arbeit

Quellenverzeichnis

Planeten und ihre Grundinformationen

ARTIKEL PLANETEN. In Wikipedia: [http://de.wikipedia.org/wiki/Planeten, 19.10.2011]
ARTIKEL MERKUR. In Wikipedia: [http://de.wikipedia.org/wiki/Merkur_%28Planet%29, 19.10.2011]
ARTIKEL VENUS. In Wikipedia: [http://de.wikipedia.org/wiki/Venus_%28Planet%29, 19.10.2011]
ARTIKEL ERDE. In Wikipedia: [http://de.wikipedia.org/wiki/Erde, 19.10.2011]
ARTIKEL MOND. In Wikipedia: [http://de.wikipedia.org/wiki/Mond, 19.10.2011]
ARTIKEL MARS. In Wikipedia: [http://de.wikipedia.org/wiki/Mars_%28Planet%29, 19.10.2011]
ARTIKEL ASTEROIDENGÜRTEL: In Wikipedia: [http://de.wikipedia.org/wiki/Asteroideng%C3%BCrtel, 19.10.2011]
ARTIKEL JUPITER. In Wikipedia: [http://de.wikipedia.org/wiki/Jupiter_%28Planet%29, 19.10.2011]
ARTIKEL SATURN. In Wikipedia:[http://de.wikipedia.org/wiki/Saturn_%28Planet%29, 19.10.2011]
ARTIKEL URANUS. In Wikipedia: [http://de.wikipedia.org/wiki/Uranus_%28Planet%29, 19.10.2011]
ARTIKEL NEPTUN. In Wikipedia: [http://de.wikipedia.org/wiki/Neptun_%28Planet%29, 19.10.2011]
ARTIKEL PLUTO. In Wikipedia: [http://de.wikipedia.org/wiki/Pluto, 19.10.2011]

Allgemeine Daten über die Planetenbahnen und deren Ephemeriden

ARTIKEL LISTE DER PLANETEN DES SONNENSYSTEMS. In Wikipedia: [http://de.wikipedia.org/wiki/Liste_der_Planeten_des_Sonnensystems]
PROF.DR.BARTELMANN, MATTHIAS; PROF.DR.HENNING, THOMAS UND DR.STAUDE, JAKOB: Sterne und Weltraum, Ahnerts Astronomisches Jahrbuch 2010. Spektrumder Wissenschaft Verlagsgesellschaft mbH. Planetenephemeriden S.180-187.

Theorie und die beinhaltenden Formeln

ARTIKEL ELLIPSE. In Wikipedia: [http://de.wikipedia.org/wiki/Ellipse, 19.10.2011]
ARTIKEL FRÜHLINGSPUNKT. In Wikipedia: [http://de.wikipedia.org/wiki/Fr%C3%BChlingspunkt, 19.10.2011]
FENDT WALTER: Zweites Keplersches Gesetz. In Java-Applets zur Physik: [http://www.walter-fendt.de/ph14d/kepler2.htm, 19.10.2011]
DR. ARNOLD, MICHAEL: Die schrittweise Berechnung von Satellitenbahnen oder Planetenbewegungen im Gravitationsfeld eines Zentralkörpers [http://www.rats-ms.de/die_faecher_stellen_sich_vor/faecher/physik/pysikalische_aufsaetze_fuer_die_sek_2/planetenbahnen_neu.pdf, 19.10.2011]
ARTIKEL VIS-VIVA-GLEICHUNG. In Wikipedia: [http://de.wikipedia.org/wiki/Vis-Viva-Gleichung, 19.10.2011]

Hilfe zur Programmierung

ORACLE®: The Java™ Tutorials. In How to Use Various Components: [http://download.oracle.com/javase/tutorial/uiswing/components/componentlist.html, 19.10.2011]

Eingefügte Bilder

Our Planet in Scale. In My[confined]Space: [http://www.myconfinedspace.com/2010/04/27/our-planet-in-scale/our-planet-in-scale-2, 21.10.2011]
UNSERE ACHT PLANETEN. In Volks- und Schulsternwarte, Planeten: [http://www.sternwarte-sohland.de/bilder/planetsmap.jpg, 19.10.2011]
MERKUR. In planeten.ch: [http://www.planeten.ch/images/plch_objekte/Merkur.gif, 19.10.2011]
VENUS. In planeten.ch: [http://www.planeten.ch/images/plch_objekte/Venus.gif, 19.10.2011]
MARS. In planeten.ch: [http://www.planeten.ch/images/plch_objekte/Mars.gif, 19.10.2011]
JUPITER. In planeten.ch: [http://www.planeten.ch/images/plch_objekte/Jupiter.gif, 19.10.2011]
SATURN. In planeten.ch: [http://www.planeten.ch/images/plch_objekte/Saturn.gif, 19.10.2011]
URANUS. In planeten.ch: [http://www.planeten.ch/images/plch_objekte/Uranus.gif, 19.10.2011]
NEPTUN. In planeten.ch: [http://www.planeten.ch/images/plch_objekte/Neptun.gif, 19.10.2011]
PLUTO. In Wikipedia: [http://de.wikipedia.org/wiki/Pluto, 19.10.2011]

Schlusswort

Zurückblickend habe ich nun, da ich die Arbeit fertiggestellt habe, ein sehr gutes Gefühl. Die Arbeit ist gelungen und das Ziel wurde erreicht. Das Programm läuft einwandfrei und zeigt das auf, was es aufzeigen sollte, nämlich die Planeten und ihre realitätsgetreuen Umlaufbahnen.

Durch die ganze Arbeit hindurch hat mich Hr. Dipl. Math. Markus Leisibach begleitet und mich auf den rechten Weg gewiesen. Für das will ich ihm nochmals danken.

Auch möchte ich mich nochmals bei Hr. Dipl. Phys. Peter Strickler herzlich bedanken. Durch seine Ideen erweiterte er mein Wissen über Planeten und Ellipsen, sowie ihren Eigenschaften.

Zuletzt möchte ich mich auch noch bei meinem Vater bedanken, der mich ebenfalls durch die ganze Arbeit begleitet hat und mir bei der Gestaltung sowohl von der schriftlichen Arbeit, als auch von der Gestaltung des Programms immer seine offene Meinung dazu gestand und Vorschläge gab, für eine bessere Möglichkeit.

Eigenständigkeitserklärung

Ich, Daniel Valério Sampaio, erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Quellen verfasst habe und ich auf eine eventuelle Mithilfe Dritter in der Arbeit ausdrücklich hinweise.