Meine komplexe Leistung

Karsten Lehmann

21. März 2016

Inhaltsverzeichnis

1 Einleitung

Die Zielstellung dieser Arbeit ist die Erstellung eines Computerspieles in der Programmiersprache Python und der Bibliothek PyGame sowie die Dokumentation des Spiels und Erstellvorganges.
Ich habe mich für dieses Projekt entschieden, da mich das Gebiet der Fachinformatik äußerst interessiert. Vor allem aber Computerspiele und ihre Faszination, die sie sowohl auf junge Leute als auch Erwachsene ausüben, begeistern mich. Deswegen habe ich mich der Herausforderung gestellt, selbst solch ein Spiel zu entwerfen. In der folgenden Arbeit kann nachvollzogen werden, welche Schritte dazu notwendig sind, ausgehend von der Programmiersprache und einfachen Anweisungen hin ein komplexes Spiel inklusive grafischer und musikalischer Gestaltung zu entwickeln.
Ich habe mich dabei für die Programmiersprache Python entschieden, da ich mit ihr bereits vertraut bin und sie eine einfache Syntax besitzt, weshalb sich Probleme mit ihr entsprechend zeitnah lösen lassen. Die Sprache bietet sowohl Anfängern als auch Fortgeschrittenen die Möglichkeit, einfache bis hin zu komplexer Software selbst zu entwickeln.
Für einen Nachbau des Spiels ,,Lander" habe ich mich entschieden, da mich vor allem ältere Computerspiele und ihre erneute Umsetzung mit ,,modernen" Mitteln interessieren. Das Spiel erschien ursprünglich 1979 und war ein sogenanntes textbasiertes Computerspiel, was bedeutet, dass es noch keine grafische Ausgestaltung gab, sondern nur Schriftzeichen im Spiel verwendet wurden.[1] Landschaften, Gegenstände, usw. wurden durch Schriftzeichen, zum Beispiel das Raute-Zeichen, Bindestriche und Ähnliches realisiert. Dies war in der damaligen Zeit für Computerspiele üblich.
Die älteste Version dieses Spiels wurde 1969 von dem Schüler Jim Storer mit 40 Zeile Code implementiert.[2] Zuerst entdeckt habe ich das Spiel auf der Uni-Website von Colorado.[3] Dort wurde das Spiel auf der Programmierplattform Adobe Flash nach gebaut und konnte direkt im Browser gespielt werden. Die Idee hinter dem Spiel fesselte mich sofort und somit beschloss ich, diese in den Mittelpunkt meiner komplexen Leistung zu setzen. Allerdings sollte das Spiel von mir neu aufbereitet und mit einer anderen, selbst gestalteten Grafik versehen werden.
Meine erste Programmierschritte waren das Erstellen einer grundlegenden, realistischen Physik der Mondlandefähre, welche das Hauptelement des Spiels darstellt. Darauf aufbauend habe ich das Spielprinzip weiter entwickelt und ein benutzerfreundliches Menü hinzugefügt. Die exakte Vorgehensweise wird im Kapitel Erstellungsverlauf näher erläutert. Im weiteren Verlauf dieser Arbeit wird zunächst über die Programmiersprache Python und deren Bibliothek PyGame berichtet, danach wird das Spielprinzip erläutert und anschließend der Erstellungsverlauf beleuchtet, gefolgt von einem Ausblick auf die Physik des Spiels und die Erstellung der fertigen, ausführbaren Version. Zum Schluss folgt eine Reflexion des Ganzen.

2 Verwendete Mittel

2.1 Python

Python ist eine freie, offene und objektorientierte Programmiersprache der Python Software Foundation. Wegen der dynamischen Typisierung zeichnet sich die Sprache durch ihre Einfachheit und Übersichtlichkeit aus. Außerdem hat sie eine einfache Syntax und umfasst bereits in der Standardinstallation eine Vielzahl

von Modulen. Weiterhin ist die Community hinter Python sehr groß, weswegen sich im Internet schnell Lösungen zu vielen gängigen Problemen finden lassen. Und schließlich muss Pythoncode nicht erst compiliert (= in Maschinensprache übersetzt) werden, sondern wird zur Laufzeit interpretiert und kann somit direkt ausgeführt und getestet werden. Außerdem bricht der Interpreter die Ausführung im Fehlerfall nicht einfach ab, sondern wirft eine Ausnahme und erst wenn diese nicht aufgefangen wird, wird sie als Fehler mit Zeilenangabe im Code ausgegeben. Aus all diesem folgt, dass die Entwicklung mit Python sehr produktiv, zeitsparend und einfach ist.[4]

Python enthält einige syntaktische Besonderheiten, so dienen zum Beispiel for-Schleifen in Python im Gegensatz zu anderen Programmiersprachen zum Iterieren (schrittweises Durchgehen von Objekten in einer Liste).
for-Schleife in C:

```
for (int i = 0; i < 20; i++)
{
        printf("i : %d\n", i);
}
```

Python-Equivalent:

```
for i in range(20):
        print("i : %d" % i)
```

Hier wird eine Liste mit den Zahlen von 0 bis 19 erzeugt, deren Elemente einzeln durchgegangen werden. Weiterhin werden Blöcke in Python durch Einrücken markiert und nicht wie in C durch Klammern oder die Schlüsselwörter “BEGIN” und “END” in Pascal.
Außerdem verwendet Python eine dynamische Typisierung, d.h. es muss bei Funktionen zum Beispiel nicht der Datentyp des Rückgabewertes oder der einzelnen Parameter angegeben werden, sondern dieser wird vom Interpreter zur Laufzeit des Programms ermittelt und entsprechend Speicherplatz reserviert.
Funktion zur Multiplikation zweier ganzer Zahlen in C:

```
int Multiplikation(int a, int b)
{
        return a * b;
}
```

Funktion zur Multiplikation zweier Objekte in Python:

```
def Multiplikation(a,b):
    return a * b
```

Die Python-Funktion ist somit auch in der Lage, Objekte verschiedenen Typs miteinander zu multiplizieren. Zum Beispiel lässt sich auch eine Zeichenkette mit einer ganzen Zahl multiplizieren.

```
print(Multiplikation("Text", 4))
Ausgabe : TextTextTextText
```

In C müsste man dafür eine eigene Funktion schreiben. Somit lässt sich Pythoncode öfters wiederverwerten. Zusätzlich können in C Speicherüberläufe nur schwer erkannt und lokalisiert werden, wie das folgende Beispiel verdeutlichen soll:

```
int main() {
    int a[10];
    int i = 0;
    while (i <= 10) {
        a[++i] = 0;
    }
    return 0;
}
```

Hier soll eigentlich nur das Feld ,,a" mit Nullen aufgefüllt werden. Wenn man dieses Programm allerdings ausführt, läuft es endlos. Um dem auf den Grund zu gehen analysiere ich den dahinter liegenden Maschinencode der Funktion ,,main":

```
00 push   %rbp
01 mov    %rsp,%rbp
02 movl   $0x0,-0x4(%rbp)
03 jmp    <08>
04 addl   $0x1,-0x4(%rbp)
05 mov    -0x4(%rbp),%eax
06 cltq
07 movl   $0x0,-0x30(%rbp,%rax,4)
08 cmpl   $0xa,-0x4(%rbp)
09 jle    <04>
10 mov    $0x0,%eax
11 pop    %rbp
12 retq
```

Wenn man sich nun den Aufbau des Stapelspeichers anschaut, erkennt man, dass der Compiler vom Base Pointer aus die ersten 4 Bytes nach unten für i reserviert hat, anschließend 4 Bytes frei lässt und danach 40 Bytes von $rbp - 8$ bis $rbp - 48$ für a folgen. Wenn man nun noch einmal das C-Programm betrachtet, fallen einem 2 Fehler ins Auge, zum einem müsste die Bedingung der while-Schleife $i < 10$ lauten und zum anderen muss die nächste Zeile ,,a[i++] = 0;" lauten, da man mit ,,++i" von 1 bis 10 statt von 0 bis 9 zählen würde.
Somit schreibt das Programm in Speicher, welcher nicht dafür vorgesehen wurde. Der endlose Programmablauf rührt nun daher, dass die Schleife die Zahlen a[1] bis a[11] mit Nullen überschreibt und a[11] dem Speicherplatz von i entspricht. In Python kann so ein Fehler nicht passieren, da er vom Interpreter erkannt werden würde. Das folgende Beispiel verdeutlicht dies:

```
a = []
```

```
for i in range(10):
    a.append(object())

a[10] = 0
```

Das Programm würde hier mit ,,IndexError: list assignment index out of range" abbrechen. Allerdings sind Pythonprogramme durch diese Überprüfungen auch deutlicher langsamer als C-Programme.

2.2 PyGame

PyGame ist eine Sammlung von Pythonmodulen, welche verschiedene Werkzeuge zur Erstellung von Computerspielen bereithalten, so zum Beispiel Funktionen und Klassen zur grafischen Ausgabe, Ausgabe von Tönen, Verarbeitung von Benutzereingaben und Bildverarbeitung. Sie basiert auf der SDL-Bibliothek und ist freie Software. Weiterhin ist es sehr portabel und läuft auf nahezu allen Plattformen und Betriebssystemen, so zum Beispiel sowohl unter Linux auf dem ARMv6 Prozessors des Raspberry Pis als auch unter Windows auf X86 und X86_64 Prozessoren. Außerdem ist PyGame freie und kostenlose Software und steht somit Jedem zur Verwendung zur Verfügung.

3 Spielprinzip

Ziel des Spieles ist es, mit der dargestellten Mondlandefähre so oft wie möglich auf dem Mond zu landen um dort geologische Proben zu entnehmen. Zunächst wird das Spiel, welches im Programmordner direkt unter dem Namen ,,Lander.exe" abgespeichert ist, gestartet. Ein Menü öffnet sich, welches die Punkte ,,Spielen, Info, Sound, Highscores, Beenden" enthält. Der Eintrag ,,Spielen" ist bereits ausgewählt und kann einfach mit der Eingabetaste bestätigt werden. Nun öffnet sich das Spielgeschehen. Man sieht nun zunächst die Landefähre vor dem Sternenhimmel. Rechts oben in der Ecke befindet sich eine Anzeige, welche die Höhe der Landefähre über der Mondoberfläche wiedergibt. Außerdem lässt sich hier die Geschwindigkeit dieser und der Füllstand des Tanks ablesen. Sollte der Tank leer sein, ist die Landefähre manövrierunfähig und das Spiel somit beendet. Links oben lässt sich die Anzahl der erfolgreichen Landungen ablesen, welche später den Highscore des Spieles bestimmt. Auf der Mondoberfläche befinden sich verschiedenartige Steine, welche nicht berührt werden sollten, da sonst die Raumsonde beschädigt wird und das Spiel scheitert.
Wichtig für die erfolgreiche Steuerung des Spieles ist es, dass die Raumsonde möglichst aufrecht landet, um nicht beschädigt zu werden. Außerdem darf die Sinkgeschwindigkeit bei der Landung $6\,\frac{m}{s}$ nicht überschreiten. Dabei müssen die Landezonen, also die Flächen auf denen die Raumsonde den Mondboden berührt, immer einen gewissen Mindestabstand voneinander haben. Die Steuerung des Spiels ist relativ einfach. Mit der linken und rechten Pfeiltaste wird die Raumsonde nach links und rechts gedreht. Durch Drücken der Leertaste lässt sich das Haupttriebwerk auf volle Leistung bringen. Das bedeutet, dass die Raumsonde mit voller Schubkraft in die gewünschte Richtung schießen kann. Mit den Pfeiltasten oben/unten kann der Schub hinsichtlich seiner Stärke kontrolliert werden. Mit der Escape-Taste kann ins Hauptmenü zurückgekehrt werden.

Nach erfolgreicher Landung kann erneut gestartet und woanders somit eine neue Probe genommen werden. Dabei wird die Landestelle mit einem dunklen Fleck und der politisch neutralen Erdflagge nach Oskar Pernefeldt (blauer Grund, 7 weiße Ringe)[5] markiert. Wie schon erwähnt, muss die nächste Landefläche dazu einen gewissen Abstand haben, sonst findet keine Markierung und somit auch keine Wertung statt. Der Spieler ist dazu motiviert eine möglichst hohe Wertung zu erreichen.
Nach Beenden des Spiels - entweder auf freiwilliger Basis oder da die Sonde kaputt gegangen ist bzw. der Treibstoff ausging - wird der Benutzer bei Erreichen eines neuen Höchstpunktestandes automatisch zur Eingabe seines Namens aufgefordert und der neue Punktestand wird abgespeichert. Wurde kein neuer Höchstpunktestand erreicht, gelangt man direkt zurück ins Hauptmenü. Die verschiedenen Punktestände können hier unter dem Menüpunkt ,,Highscores" betrachtet werden.
Der Menüpunkt ,,Info" beinhaltet eine kurze Beschreibung des Spielprinzips. Der Punkt ,,Optionen" befähigt den Nutzer, Musik und Töne des Spieles ein- und auszuschalten und mit ,,Beenden" wird das Spiel geschlossen.

4 Erstellungsverlauf

Als erstes habe ich das eigentliche Spiel programmiert. Dafür musste zunächst die Physik der Landefähre implementiert werden, damit diese auf der Oberfläche landen und hin-und-her fliegen kann. Anschließend setzte ich die Bewegung der Kamera um, damit sich die Landefähre immer in Bildmitte befindet. Somit wird verhindert, dass die Raumfähre den für den Spieler einsehbaren Bereich verlässt. Danach suchte ich nach einer Möglichkeit, den Schwierigkeitsgrad des Spiels zu steigern um den Spieler nicht zu unterfordern. Eine gute Option sah ich darin, die maximale Flugzeit durch eine fest definierte Menge an Treibstoff zu begrenzen. So ist der Spieler gezwungen, die Landefähre sparsam und zielgerichtet zu steuern und zu landen, bevor ihm der Treibstoff ausgeht. Des weiteren implementierte ich eine zufällige Verteilung von Hindernissen auf der Mondoberfläche, um die Anforderung weiter zu steigern. Die zweite große Programmieraufgabe war die Umsetzung der Menüstrukturen. Dafür habe ich zuerst das Hauptmenü mit den zwei Punkten ,,Spielen" und ,,Beenden" erstellt. Zudem bewirkte ich eine Rückkehr in dieses Menü nach Beendigung des Spiels. Als nächstes setzte ich die einzelnen Menüpunkte ,,Info", ,,Optionen" und ,,Highscore" dazu. Außerdem musste ein Dialog entstehen, falls eine neue Höchstpunktzahl erreicht wurde. In diesem Fenster trägt der Spieler dann seinen Namen ein, unter welchem der erreichte Punktestand gespeichert wird. Als letztes wurde der Menüpunkt Optionen durch den Eintrag Sound ersetzt.

5 Erstellung der Grafiken und Musik

Die Bilder des Mondes, der Mondoberfläche, der Steine und der Landezone wurden mit Blender erstellt, der Sternenhimmel, die Landefähre und die Elemente der Benutzeroberfläche mit Gimp. Außerdem wurde die im Spiel verwendete Schriftart mit FontForge erzeugt. Für die musikalische Gestaltung wurden Hydrogen und Audacity verwendet.
Blender ist eine professionelle und freie Software für 3D Computergrafiken und 3D Modellierung. Für die

Bilderstellung musste ich die einzelnen Objekte (Mondoberfläche, Steine...) erst als 3D Objekt modellieren und anschließend von dem 3D Objekt ein Bild berechnen lassen. Die Bilder wurden teilweise noch mit Gimp nach bearbeitet. Der Vorteil ist, dass sich auf diese Art und Weise mit simplen Mitteln teilweise fotorealistische Bilder umsetzen lassen.
Gimp ist ein freies Programm zur Erstellung und Bearbeitung von Grafiken am Computer. Die damit erstellten Bilder sind allesamt Hand gezeichnet, beziehungsweise durch die Kombination verschiedener Filter erzeugt.
FontForge ist ebenfalls freie Software und dient der Entwicklung und Bearbeitung von Schriftarten. Damit musste ich alle Zeichen der neuen Schriftart einzeln als Vektorgrafik erstellen und deren Abstand zum nächsten Zeichen definieren. Hydrogen ist eine freie Software und Drum-Machine. Mit diesem Programm wurde der kurze, sich wiederholende Track im Hintergrund erstellt. Die weiteren Töne sind von mir angefertigte Tonaufnahmen, welche später mit Audacity nach bearbeitet wurden. Audacity ist ein freier Audioeditor und Rekorder. Beispiele für diese Tonaufnahmen wären das Triebwerksgeräusch der Raumfähre, welches ursprünglich eine Aufnahme von einer Sprühflasche war, oder der Ton, welcher abgespielt wird, wenn man das Spiel verloren hat. Dieser ist eine Aufnahme des Radiorauschens, zu welcher später noch ein Echo-Effekt hinzugefügt wurde.

6 Physik rund um die Landefähre

Das Spiel orientiert sich an der ersten Mondlandung. Dabei besaß das Landemodul in der Realität zwei getrennte Triebwerksmodule, eins für die Landung, welches auf dem Mond verbleibt, mit einer Masse von 10246 kg (inklusive Treibstoff) und eins für den erneuten Aufstieg zum Kommando-/Service Modul, welches eine Masse von 4819 kg mit Treibstoff besitzt und die Crew beinhaltet[6]. Die maximale Schubkraft des Abstiegsmoduls beträgt 45,04 kN[7] und die des Aufstiegsmoduls beträgt 16 kN[8]. Im Spiel habe ich mich für eine maximale Schubkraft von 16 kN entschieden und mit einer Masse von 4 t gerechnet.
Neben der Schubkraft der Triebwerke wirkt im Spiel auf die Landefähre noch die Anziehungskraft des Mondes. Im Spiel wird aber nicht mit Kräften gerechnet, sondern nur mit Geschwindigkeitsänderungen. Die Landefähre besitzt einen Bewegungsvektor, dessen Betrag deren Geschwindigkeit entspricht. Für die Anziehungskraft des Mondes wird beispielsweise zur y-Koordinate des Vektors beständig die Geschwindigkeit $v = g * t = 1,62\frac{m}{s^2} * \Delta t$ addiert. Δt entspricht der Zeitdifferenz zur letzten Berechnung der Geschwindigkeit. Auch für die Schubkraft, die über einen Vektor $\vec{F}$ gegeben ist, werden permanent die Geschwindigkeitsänderungen berechnet $\vec{\Delta v} = \frac{\vec{F} * \Delta t}{m}$. Der $\vec{\Delta v}$ Vektor muss dann nur noch vom eigentlichen Geschwindigkeitsvektor abgezogen werden.
Ein weiteres Problem stellt die Tatsache dar, dass Entfernungen im Spiel im allgemeinen in Pixeln angegeben werden, während sie für Berechnungen in Metern benötigt werden. Zur Lösung dieses Problems habe ich zuerst die Höhe des Landemoduls mit 5,5m ermittelt[9], dann die Höhe im Spiel mit 40 Pixel, daraus ergibt sich ein Verhältnis von $\frac{40 Pixel}{5,5 Meter} \approx 7 \frac{Pixel}{Meter}$.

7 Algorithmus zum Verteilen von Hindernissen auf der Mondoberfläche

Die Verteilung der Hindernisse auf der Mondoberfläche wird durch die Funktion ,,gen_landscape()" aus der Datei ,,lander.py" erledigt. Die Anforderungen an diese Funktion liegen darin, dass die Hindernisse zufällig auf der Landschaft verteilt werden, sich dabei aber nicht überlappen und zwischen ihnen noch genug Platz für die Mondlandefähre zum Landen ist.
Dabei wird zuerst die maximale Anzahl an Hindernissen errechnet, die man auf der gegebenen Fläche platzieren könnte (Spielraum). Anschließend wird nun die tatsächliche Anzahl der Hindernisse zwischen der viertelsten und halben maximalen Anzahl an Hindernissen zufällig festgelegt. Danach wird Anzahl-mal solange eine zufällige Position bestimmt, bis diese nicht mit den Positionen bereits vorhandener Hindernisse kollidiert. Ist diese Bedingung erfüllt, wird ein neues Hindernis mit der ermittelten Position hinzugefügt. Der Nachteil dieses Algorithmus' ist, dass er theoretisch eine unendlich lange Laufzeit besitzen kann, für den Fall, dass der Zufallsgenerator jedes mal dieselbe, oder zumindest sehr ähnliche Zahlen zurück gibt. Der komplette Algorithmus lässt sich im folgendem Struktogramm betrachten und eine Implementation davon befindet sich in der Datei ,,lander.py" im beiliegenden Quellcode.

Landschaft erzeugen

- Spielraum = RechteGrenze - LinkeGrenze
- Spielraum = $\frac{Spielraum}{64}$
- Anzahl = ZufälligeZahl($\frac{Spielraum}{4}$, $\frac{Spielraum}{2}$)
- Hindernisse = neue Gruppe
- Wiederhole ,,Anzahl" mal
  - Überlappungen = Wahr
  - Solange Überlappungen
    - Überlappungen = Falsch
    - Position = ZufälligeZahl(LinkeGrenze, RechteGrenze)
    - Für jedes Hindernis in Gruppe
      - Wenn $Hindernis.Position - 100 < Position < Hindernis.Position + 100$:

Wahr	Falsch
Überlappung = Wahr	Mache weiter

  - Hindernis = neues Hindernis(Position)
  - Füge Hindernis zu Gruppe hinzu

8 Speichern der Höchstpunktzahlen

Die Höchstpunktzahlen (Highscores) werden im Ordner des Spieles in der Datei ,,.scores" im JSON-Format gespeichert. Dabei wird eine Liste aus 5 weiteren Listen mit jeweils dem Namen und der Punktzahl erstellt und abgespeichert. Zum Zurücksetzen der Punktzahl reicht es aus, diese Datei zu löschen.

9 Erstellung der unter Windows ausführbaren Version

Um das Programm in eine unter Windows selbstständig und von der Pythonbibliothek unabhängig ausführbare Form zu bringen habe ich das Programm Py2exe verwendet, da ich im Umgang mit diesem schon Erfahrungen in früheren Projekten gesammelt habe. Um nur unter Windows eine ausführbare Datei des Spiels zu erstellen muss man nach vorheriger Installation von Python und Py2Exe die Datei ,,setup.bat" ausführen. Alle für das Spiel benötigten Dateien befinden sich nun in dem neu entstandenen Ordner ,,dist".

10 Reflexion

Die Umsetzung des Projektes war für mich eine große Herausforderung. Trotz gegebener Vorlage im Hinblick auf die Originalversion gab es einige Schwierigkeiten, welche es im Erstellungsprozess zu bewältigen gab. Zunächst war es problematisch, dass die von Python verwendeten Bibliotheken keine Hilfsmittel zum Aufbau von Menüstrukturen enthielten. Da das Menü aber ein äußerst wichtiges Element von Computerspielen darstellt, musste dafür eine Lösung gefunden werden. Ich musste also eine eigene Bibliothek erstellen. Dazu musste ich zunächst analysieren, welche Objekte für mein Menü benötigt wurden und diese dann implementieren. Dies nahm einige Zeit in Anspruch, konnte aber letztendlich von mir gelöst werden.
Im Nachhinein war die Wahl der Programmiersprache Python eine gute Entscheidung, da sich mit dieser äußerst produktiv arbeiten lässt und ich mich somit mehr auf die anderen Probleme der Arbeit konzentrieren konnte. Als weiteres Projekt würde ich aber auch gern den Versuch wagen, dieses oder ein anderes Spiel in einer niedrigeren Programmiersprache umzusetzen. Beispielsweise mit der Sprache C, was allerdings einen erheblichen Mehraufwand an Zeit und einige Kenntnisse benötigen würde, welche ich mir noch erarbeiten muss.
Zusammengefasst bin ich mit dem Ergebnis meines Projekts sehr zufrieden. Es wurde sowohl von mir als auch von anderen, mit diesem Spieltyp nicht vertrauten Personen getestet und als sehr unterhaltsam empfunden.

Literatur

[1] `https://en.wikipedia.org/wiki/Lunar_Lander_%281979_video_game%29` 20.02.2016 17:00

[2] `http://www.atarimagazines.com/compute/issue166/90_Blast_off.php` 20.02.2016 18:00

[3] https://phet.colorado.edu/sims/lunar-lander/lunar-lander_en.html 20.02.2016 18:00

[4] https://www.python.org/doc/essays/blurb/ 19.03.2016 17:30

[5] https://en.wikipedia.org/wiki/Flag_of_the_Earth 15.10.2015 16:30

[6] http://nssdc.gsfc.nasa.gov/nmc/spacecraftDisplay.do?id=1969-059C 23.02.2016 18:00

[7] https://en.wikipedia.org/wiki/Descent_Propulsion_System 23.02.2016 18:00

[8] https://en.wikipedia.org/wiki/Ascent_Propulsion_System 23.02.2016 18:00

[9] https://en.wikipedia.org/wiki/Apollo_Lunar_Module 23.02.2016 18:00