

Краткое описание GRASS Graphical Modeler

[Обсудить в форуме](#) Комментариев — 28

Эта страница опубликована в основном списке статей сайта по адресу <http://gis-lab.info/qa/grass-modeller.html>

Описание расширения GRASS для визуального моделирования и материалы для самостоятельной работы

[GRASS Graphical Modeler](#) - это расширение GRASS, которое позволяет пользователю создавать, редактировать, управлять и выполнять моделями геопространственного анализа. GRASS Graphical Modeler написана командой разработчиков GRASS. Документация была создана [Мartiном Ланда](#). Начиная с версии GRASS 6.4.2 это расширение включено в программу по умолчанию, отдельная установка не требуется.

На момент написания статьи расширение находилось в стадии разработки.

Содержание

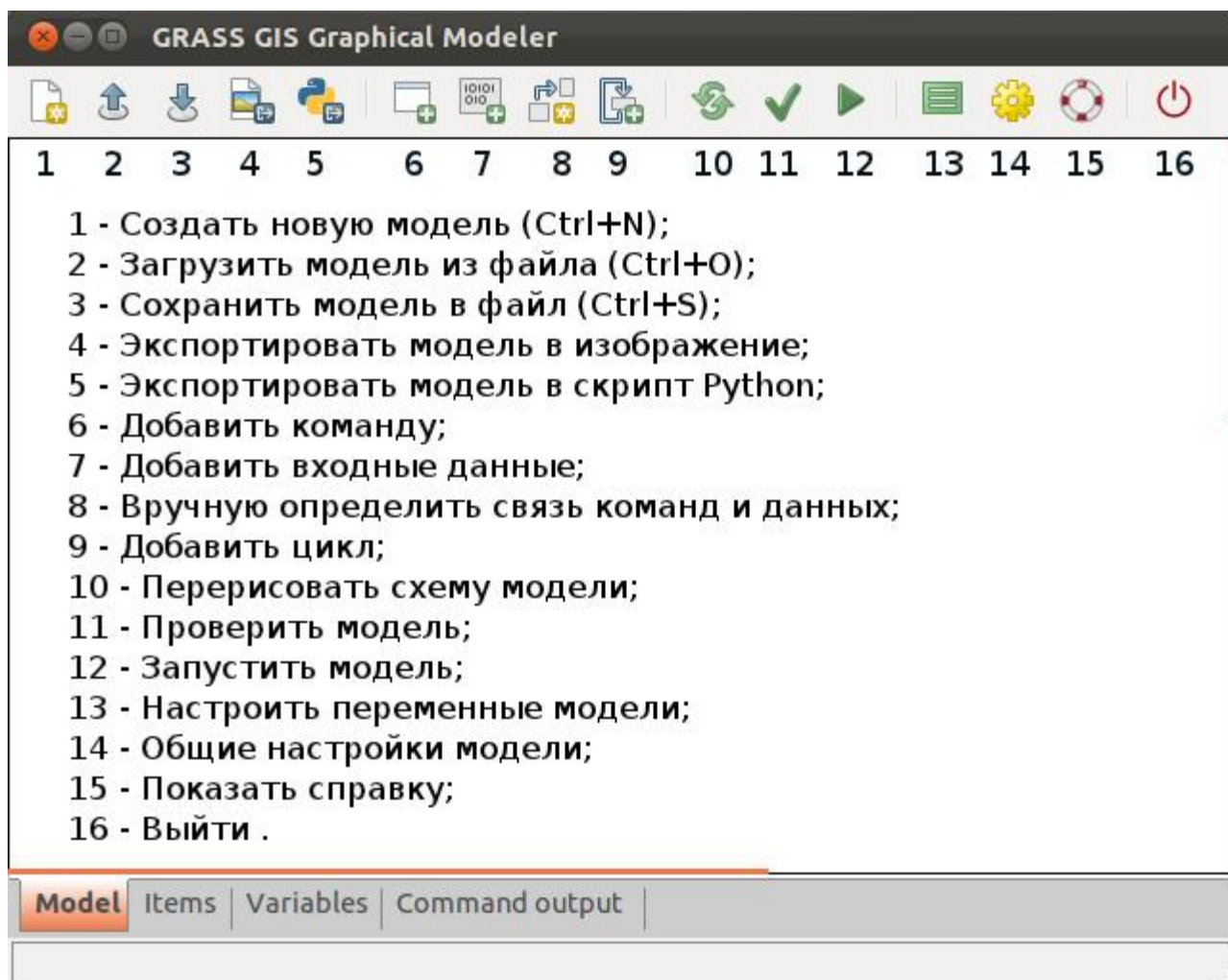
- [1 Введение](#)
- [2 Задача](#)
 - [2.1 Цель](#)
 - [2.2 Инструменты и данные](#)
- [3 Алгоритм действий](#)
 - [3.1 Добавляем команды](#)
 - [3.2 Задаём переменные](#)
 - [3.3 Реорганизуем блок-схему](#)
 - [3.4 Проверяем и запускаем модель](#)
- [4 Экспорт в Python](#)
- [5 Выводы](#)
- [6 Ссылки](#)

Введение

Расширение wxGUI Modeler в некоторой степени напоминает [ArcGIS ModelBuilder](#), который в свою очередь, похож на [ERDAS IMAGINE Spatial Modeler](#). Согласно Википедии, Spatial Modeler впервые появилась в 1993 году. После этого в 2004 году ESRI (Environmental Systems Research Institute) создали собственный инструмент под названием ModelBuilder, [сообщает Википедия](#).

Расширение позволяет выполнять следующее:

- задавать действия (команды GRASS);
- задавать входящие данные (растровые, векторные и 3D-растровые);
- устанавливать связи между действиями и входными данными;
- задавать циклы и условия выполнения;
- проверять модель на работоспособность;
- запускать модель;
- сохранять настройки модели в файл (*.gxm);
- экспортировать модель в скрипт на Python'e;
- экспортировать концептуальную модель в изображение (поддерживаемые форматы: PNG, BMP, GIF, JPG, PCX, PNM, TIF, XPM).



Работа расширения GRASS wxGUI Modeler рассматривается на примере создания серии композитных изображений по данным LANDSAT 7. Если неохота искать и качать сцены с сайта [USGS](https://usgs.gov/), то можно потренироваться на [доступных данных](#) на этом сайте.

Задача

Часто нужно сделать композитные изображения различных каналов LANDSAT. Обычно этот процесс разделён на несколько этапов:

1. Импорт растров в GRASS (модуль `r.in.gdal`);
2. Атмосферная коррекция для устранения влияния атмосферы (модуль `i.landsat.toar`);
3. Автоматическое улучшение цветовой карты (модуль `i.landsat.rgb`);
4. Создание композитного изображения (модуль `r.composite`).

Цель

Автоматически создавать скорректированные по атмосфере, композитные изображения следующих комбинаций каналов: 321, 453, 543, 742, 745, 754.

Инструменты и данные


- Операционная система Ubuntu 11.10
- GRASS 7.0.svn50461 (2012)
- Любой набор снимков LANDSAT (сенсор ETM+)

Алгоритм действий

Предположим, что растры уже импортированы в GRASS, район GRASS установлен корректно и метод, используемый при атмосферной коррекции - "uncorrected".

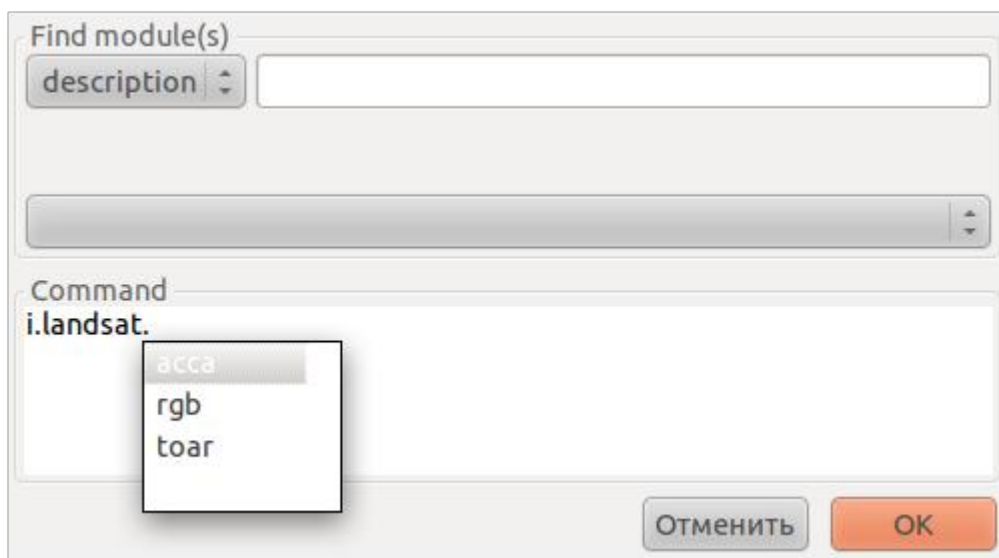
1. Переименовать растры по шаблону B.* Например, xxxxxxxxxxxxxx_B10 → B.1 (можно воспользоваться

[скриптом](#) для переименования);

2. Запустить расширение через меню File → Graphical Modeler или кликая на иконку 
3. Добавить команды, необходимые для решения поставленной задачи
 - i.landsat.toar -t --overwrite --verbose input_prefix=B. output_prefix=toar. metfile=%metfile sensor=tm7 method=uncorrected percent=0.01 pixel=1000 sat_zenith=8.2000 rayleigh=0.0
 - i.landsat.rgb --verbose red=toar.3 green=toar.2 blue=toar.1 strength=98
 - r.composite --overwrite --verbose red=toar.3 green=toar.2 blue=toar.1 levels=32 output=321
 - i.landsat.rgb --verbose red=toar.4 green=toar.5 blue=toar.3 strength=98
 - r.composite --overwrite --verbose red=toar.4 green=toar.5 blue=toar.3 levels=32 output=453
 - i.landsat.rgb --verbose red=toar.5 green=toar.4 blue=toar.3 strength=98
 - r.composite --overwrite --verbose red=toar.5 green=toar.4 blue=toar.3 levels=32 output=543
 - i.landsat.rgb --verbose red=toar.7 green=toar.4 blue=toar.2 strength=98
 - r.composite --overwrite --verbose red=toar.7 green=toar.4 blue=toar.2 levels=32 output=742
 - i.landsat.rgb --verbose red=toar.7 green=toar.4 blue=toar.5 strength=98
 - r.composite --overwrite --verbose red=toar.7 green=toar.4 blue=toar.5 levels=32 output=745
 - i.landsat.rgb --verbose red=toar.7 green=toar.5 blue=toar.4 strength=98
 - r.composite --overwrite --verbose red=toar.7 green=toar.5 blue=toar.4 levels=32 output=754
1. Задать переменные (в нашем случае она одна - %metfile)
2. Проверить модель на работоспособность
3. Запустить модель

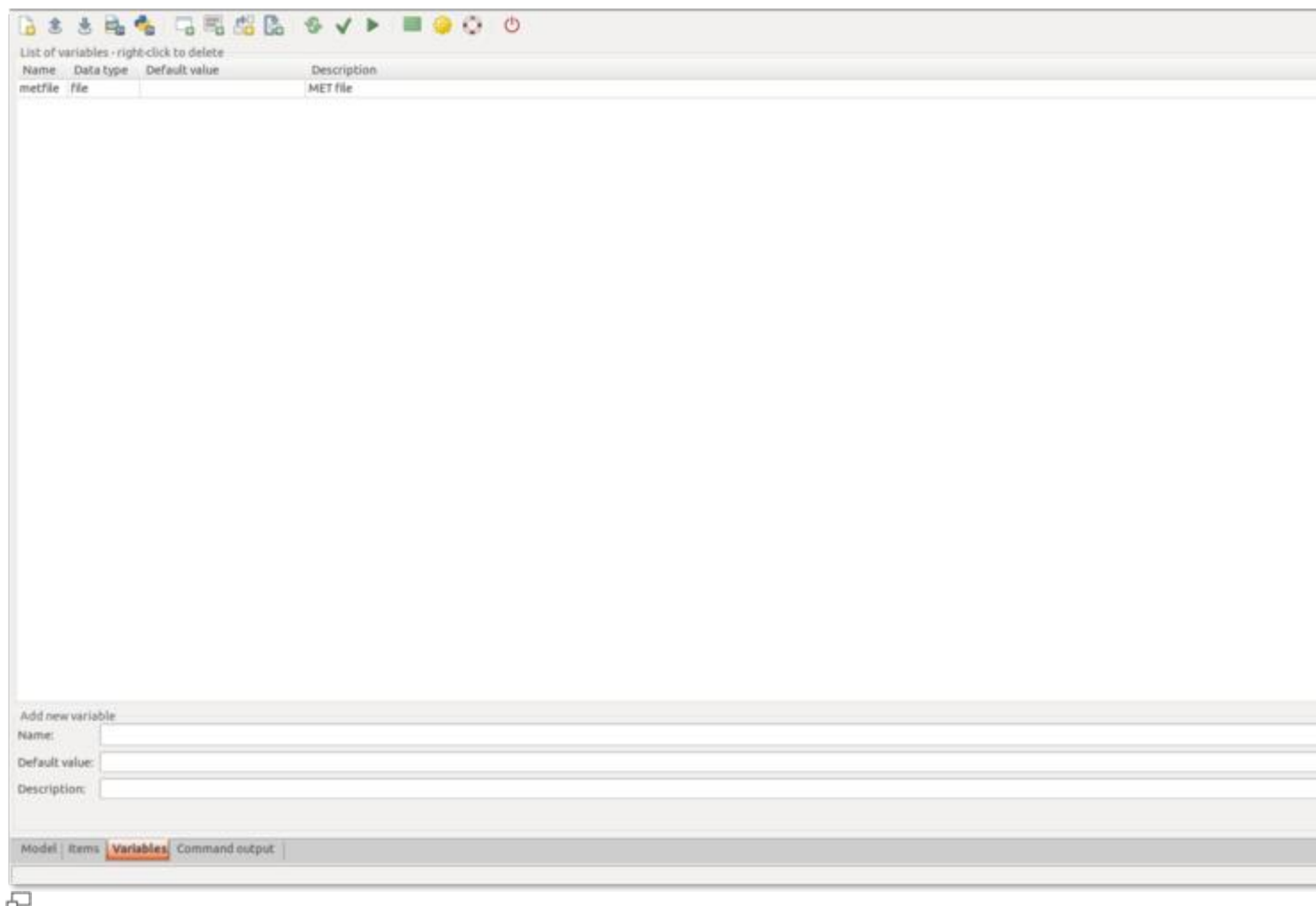
Добавляем команды

Прежде всего, следует начать с добавления новых команд (меню Model -> Add command).



Задаём переменные

Переменные задаются знаком %, т.е. переменная *%metfile* обозначает название заголовка файла (.met / MTL.txt) для сенсоров Landsat ETM+ или TM5. Переменные могут быть следующих типов: строковые, целые числа, числа с плавающей точкой, векторные, растровые, набор данных или путь к файлу. Все переменные, используемые в модели, должны быть добавлены на вкладке Variables.



Вкладка Variables, позволяющая управлять переменными в GRASS Graphical Modeler

Таким образом, первая команда будет выглядеть так:

```
i.landsat.toar -t --overwrite --verbose input_prefix=B. output_prefix=toar. metfile=%metfile sensor=tm7
method=uncorrected percent=0.01 pixel=1000 sat_zenith=8.2000 rayleigh=0.0
```

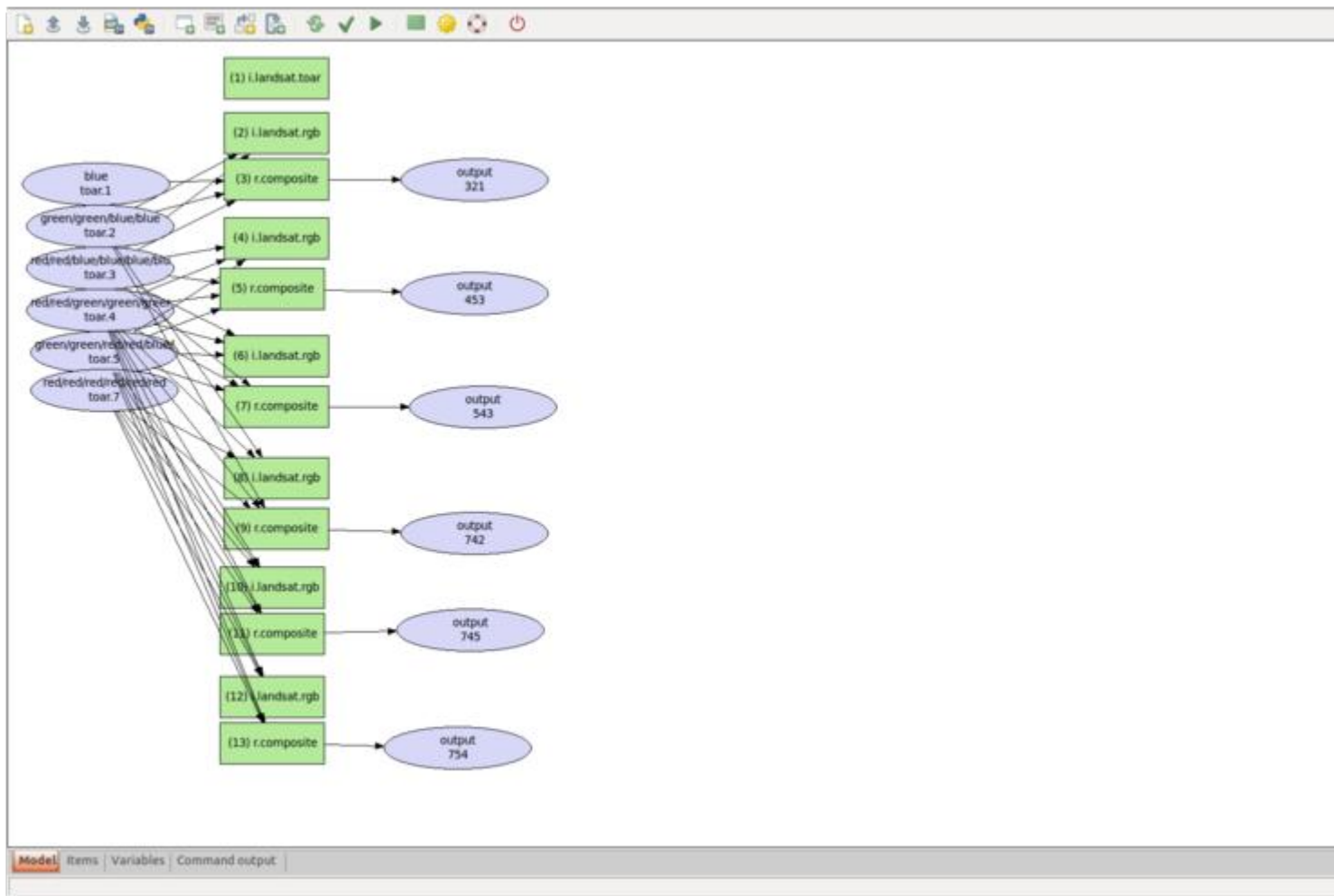
В результате мы должны получить список команд, которые будут выполнены автоматически при запуске модели.

List of items - right-click to delete			
ID	Name	In block	Command / Condition
1	i.landsat.toar		i.landsat.toar -t -overwrite -verbose input_prefix=B. output_prefix=toar. metfile=%metfile sensor=tm7 method=uncorrected percent=0.01 pixel=1000 sat_zenith=8.2000 rayleigh=0.0
2	i.landsat.rgb		i.landsat.rgb -verbose red=toar.3 green=toar.2 blue=toar.1 strength=98
3	r.composite		r.composite -overwrite -verbose red=toar.3 green=toar.2 blue=toar.1 levels=32 output=321
4	i.landsat.rgb		i.landsat.rgb -verbose red=toar.4 green=toar.5 blue=toar.3 strength=98
5	r.composite		r.composite -overwrite -verbose red=toar.4 green=toar.5 blue=toar.3 levels=32 output=453
6	i.landsat.rgb		i.landsat.rgb -verbose red=toar.5 green=toar.4 blue=toar.3 strength=98
7	r.composite		r.composite -overwrite -verbose red=toar.5 green=toar.4 blue=toar.3 levels=32 output=543
8	i.landsat.rgb		i.landsat.rgb -verbose red=toar.7 green=toar.4 blue=toar.2 strength=98
9	r.composite		r.composite -overwrite -verbose red=toar.7 green=toar.4 blue=toar.2 levels=32 output=742
10	i.landsat.rgb		i.landsat.rgb -verbose red=toar.7 green=toar.4 blue=toar.5 strength=98
11	r.composite		r.composite -overwrite -verbose red=toar.7 green=toar.4 blue=toar.5 levels=32 output=745
12	i.landsat.rgb		i.landsat.rgb -verbose red=toar.7 green=toar.5 blue=toar.4 strength=98
13	r.composite		r.composite -overwrite -verbose red=toar.7 green=toar.5 blue=toar.4 levels=32 output=754



Список команд, которые будут выполнены автоматически при запуске модели в wxGUI Modeler

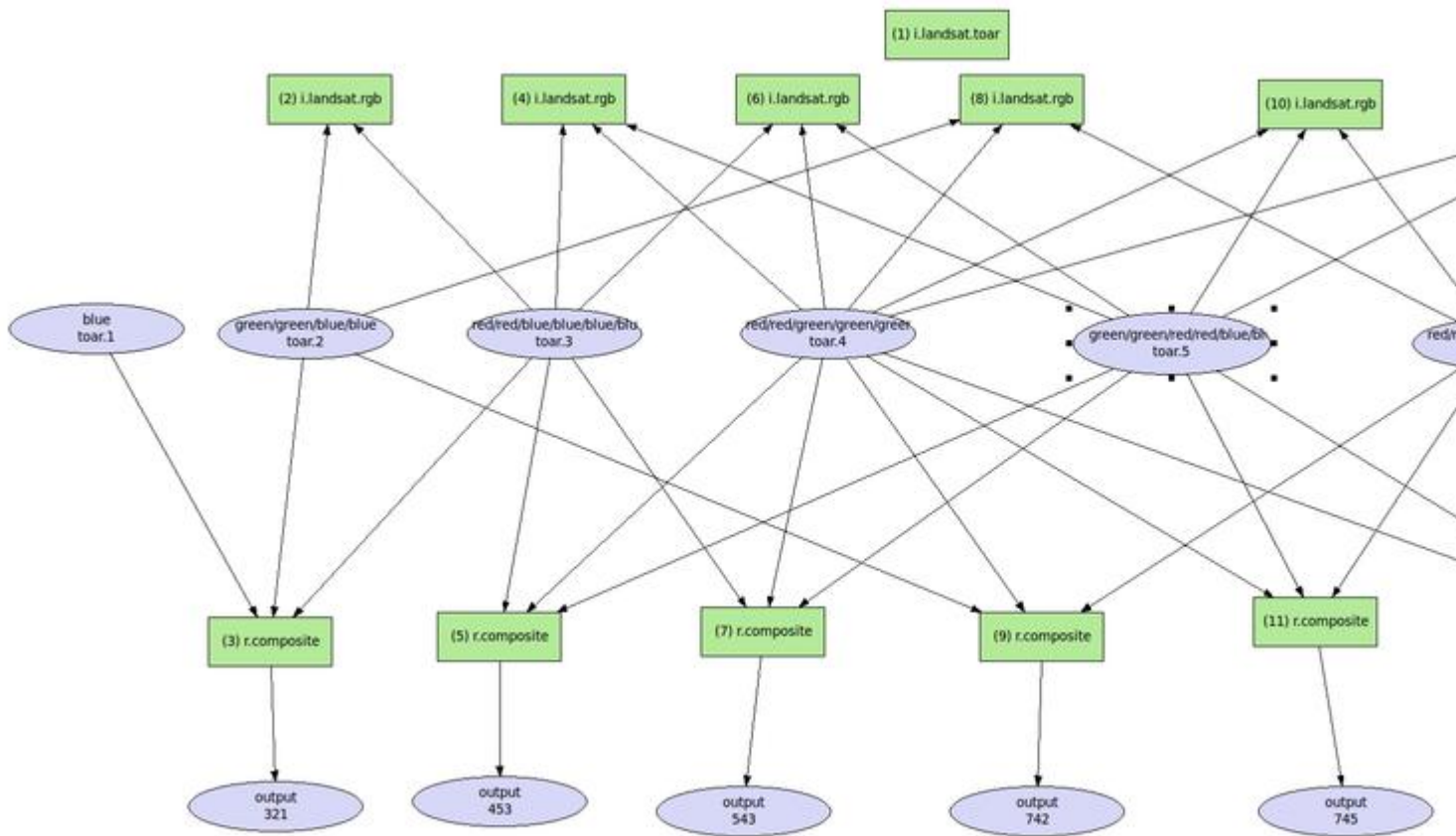
Если вернуться к вкладке Model, то можно увидеть блок-схему всего процесса.



Изначальная блок-схема в окне wxGUI Modeler

Реорганизуем блок-схему

Я советую реорганизовать схему так, чтобы она выглядела более или менее понятной для тех, кто будет использовать модель позже.



Блок-схема, приведённая в нормальный вид

Проверяем и запускаем модель

И наконец, следует проверить модель и запустить её!

```

at-sensor radiance = 0.62163 * DN + -5.62165
mean solar exoatmospheric irradiance (E0DN): 1551.000
at-sensor reflectance = radiance / 229.60635

-----
BAND 4 (code 4)
calibrated digital number (DN): 1.0 to 255.0
calibration constants (L): -5.100 to 157.400
at-sensor radiance = 0.63976 * DN + -5.73976
mean solar exoatmospheric irradiance (E0DN): 1044.000
at-sensor reflectance = radiance / 154.55127

-----
BAND 5 (code 5)
calibrated digital number (DN): 1.0 to 255.0
calibration constants (L): -1.000 to 31.040
at-sensor radiance = 0.12622 * DN + -1.12622
mean solar exoatmospheric irradiance (E0DN): 225.700
at-sensor reflectance = radiance / 33.41209

-----
BAND 6 thermal (code 61)
calibrated digital number (DN): 1.0 to 255.0
calibration constants (L): 0.000 to 17.040
at-sensor radiance = 0.06709 * DN + -0.06709
at-sensor temperature = 1282.710 / log((664.090 /
radiance) + 1.0)

-----
BAND 6 thermal (code 62)
calibrated digital number (DN): 1.0 to 255.0
calibration constants (L): 3.200 to 12.600
at-sensor radiance = 0.03720 * DN + 3.16280
at-sensor temperature = 1282.710 / log((664.090 /
radiance) + 1.0)

-----
BAND 7 (code 7)
calibrated digital number (DN): 1.0 to 255.0
calibration constants (L): -0.350 to 10.800
at-sensor radiance = 0.04390 * DN + -0.39390
mean solar exoatmospheric irradiance (E0DN): 82.070
at-sensor reflectance = radiance / 12.14945

-----
BAND 8 (code 8)
calibrated digital number (DN): 1.0 to 255.0
calibration constants (L): -4.700 to 243.100
at-sensor radiance = 0.97559 * DN + -5.67559
mean solar exoatmospheric irradiance (E0DN): 1364.000
at-sensor reflectance = radiance / 202.51546

Calculating...
Writing reflectance of <B.1> to <toar.1>...
Writing reflectance of <B.2> to <toar.2>...
Writing reflectance of <B.3> to <toar.3>...
Writing reflectance of <B.4> to <toar.4>...
Writing reflectance of <B.5> to <toar.5>...
Writing temperature of <B.61> to <toar.61>...
Writing temperature of <B.62> to <toar.62>...
Writing reflectance of <B.7> to <toar.7>...
Writing reflectance of <B.8> to <toar.8>...

```

Output window

Очистить Сохранить

Command prompt

Остановить

Model Rems Variables **Command output**

13 items (13 actions) loaded into model

Экспорт в Python

Впоследствии возможно сделать экспорт модели в скрипт Python для более тонкой доработки модели. Вид первоначального скрипта имеет вид:

```
#!/usr/bin/env python
#
#####
#
# MODULE:          model_atcor7_composite
#
# AUTHOR(S):       Vladimir Naumov
#
# PURPOSE:         Script generated by wxGUI Graphical Modeler.
#
# DATE:           Tue Jul 24 09:32:51 2012
#
#####

import sys
import os
import atexit

import grass.script as grass

def cleanup():
    pass

def main():
    grass.run_command("i.landsat.toar",
                      flags = 't',
                      overwrite = True,
                      verbose = True,
                      input_prefix = "B.",
                      output_prefix = "toar.",
                      metfile = metfile,
                      sensor = "tm7",
                      method = "uncorrected",
                      percent = 0.01,
                      pixel = 1000,
                      sat_zenith = 8.2000,
                      rayleigh = 0.0)
    grass.run_command("i.landsat.rgb",
                      verbose = True,
                      red = "toar.3",
                      green = "toar.2",
                      blue = "toar.1",
                      strength = 98)
    grass.run_command("r.composite",
                      overwrite = True,
                      verbose = True,
                      red = "toar.3",
                      green = "toar.2",
                      blue = "toar.1",
                      levels = 32,
                      output = "321")
    grass.run_command("i.landsat.rgb",
                      verbose = True,
                      red = "toar.4",
                      green = "toar.5",
                      blue = "toar.3",
                      strength = 98)
    grass.run_command("r.composite",
                      overwrite = True,
```



```

        verbose = True,
        red = "toar.4",
        green = "toar.5",
        blue = "toar.3",
        levels = 32,
        output = "453")
grass.run_command("i.landsat.rgb",
        verbose = True,
        red = "toar.5",
        green = "toar.4",
        blue = "toar.3",
        strength = 98)
grass.run_command("r.composite",
        overwrite = True,
        verbose = True,
        red = "toar.5",
        green = "toar.4",
        blue = "toar.3",
        levels = 32,
        output = "543")
grass.run_command("i.landsat.rgb",
        verbose = True,
        red = "toar.7",
        green = "toar.4",
        blue = "toar.2",
        strength = 98)
grass.run_command("r.composite",
        overwrite = True,
        verbose = True,
        red = "toar.7",
        green = "toar.4",
        blue = "toar.2",
        levels = 32,
        output = "742")
grass.run_command("i.landsat.rgb",
        verbose = True,
        red = "toar.7",
        green = "toar.4",
        blue = "toar.5",
        strength = 98)
grass.run_command("r.composite",
        overwrite = True,
        verbose = True,
        red = "toar.7",
        green = "toar.4",
        blue = "toar.5",
        levels = 32,
        output = "745")
grass.run_command("i.landsat.rgb",
        verbose = True,
        red = "toar.7",
        green = "toar.5",
        blue = "toar.4",
        strength = 98)
rass.run_command("r.composite",
        overwrite = True,
        verbose = True,
        red = "toar.7",
        green = "toar.5",
        blue = "toar.4",
        levels = 32,
        output = "754")

```

```

return 0

```

```

if __name__ == "__main__":
    options, flags = grass.parser()
    atexit.register(cleanup)
    sys.exit(main())

```

Зная язык программирования [Python](#), первоначальный скрипт можно привести к более лаконичному виду и использовать его уже в повседневной работе:

```

#!/usr/bin/env python
import sys
import os
import atexit

import grass.script as grass

def cleanup():
    pass

def main():
    grass.run_command("i.landsat.toar",
                      flags='t',
                      overwrite=True,
                      verbose=True,
                      input_prefix="B.",
                      output_prefix="toar.",
                      metfile=metfile,
                      sensor="tm7",
                      method="uncorrected",
                      percent=0.01,
                      pixel=1000,
                      sat_zenith=8.2000,
                      rayleigh=0.0)
    list = ["321", "453", "543", "742", "745", "754"]
    for band in list:
        grass.run_command("i.landsat.rgb",
                          red="toar."+band[0],
                          green="toar."+band[1],
                          blue="toar."+band[2],
                          strength=98)

        grass.run_command("r.composite",
                          red="toar."+band[0],
                          green="toar."+band[1],
                          blue="toar."+band[2],
                          levels=32,
                          output=band)

    return 0

if __name__ == "__main__":
    options, flags = grass.parser()
    atexit.register(cleanup)
    sys.exit(main())

```

Выводы

В результате запуска модели, шесть растров были готовы менее, чем за десять минут.



Коллаж из растров, созданных в результате работы GRASS wxGUI Modeler.

Время начала запуска: 9:34:16; время окончания работы модели: 9:41:35. Кроме того, помимо собственно композитных изображений были созданы скорректированные по атмосфере растры всех каналов, которые могут быть использованы для дальнейшего анализа.

[Файл](#) модели и скрипт переименования каналов LANDSAT могут быть использованы для самостоятельного запуска модели.

Ссылки

Источник: [Vladimir Naumov – Graphical Modeler in GRASS](#)

[Официальная страница GRASS Graphical Modeler](#)

[Справка ArcGIS ModelBuilder](#)

[ERDAS IMAGINE Spatial Modeler \(википедия\)](#)

[Файл:L7 toar composite.zip](#)

[Обсудить в форуме](#) Комментариев — 28

Последнее обновление: 2014-05-15 00:42

Дата создания: 24.06.2012

Автор(ы): [Владимир Наумов](#)