Software Design for Astronomers

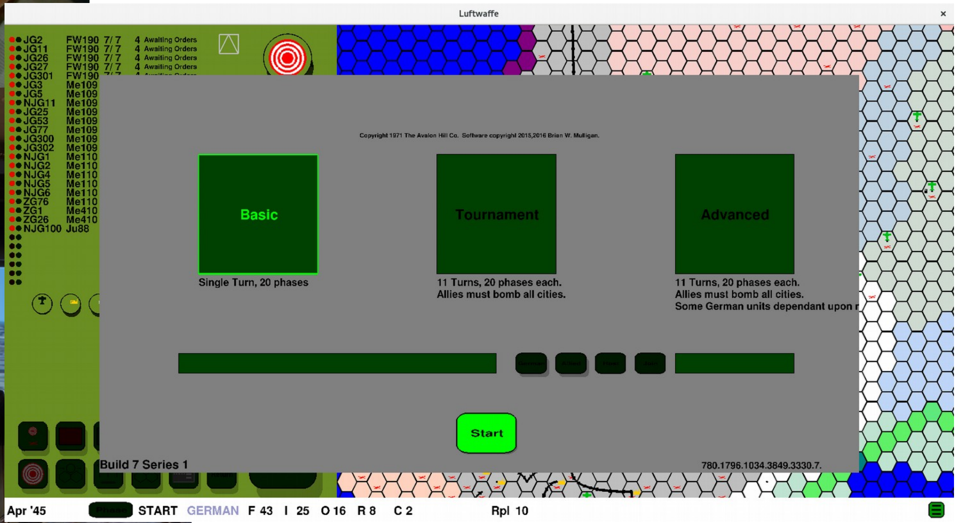
Brian W. Mulligan
University of Texas at Austin

Grad Student – Post-Doc Seminar
University of Texas at Austin
15 Sept. 2017

My Bio

```
EXTENDED COLOR BASIC 1.1
COPYRIGHT (C) 1982 BY TANDY
UNDER LICENSE FROM MICROSOFT
OK
```

MINNESOTA STATE UNIVERSITY
MAVERICKS



```
JG2     FW190 7/ 7   4 Awaiting Orders
JG11    FW190 7/ 7   4 Awaiting Orders
JG26    FW190 7/ 7   4 Awaiting Orders
JG27    FW190 7/ 7   4 Awaiting Orders
JG301   FW190
JG3     Me109
JG5     Me109
NJG11   Me109
JG25    Me109
JG53    Me109
JG77    Me109
JG300   Me109
JG302   Me109
NJG1    Me110
NJG2    Me110
NJG4    Me110
NJG5    Me110
NJG6    Me110
ZG76    Me110
ZG1     Me410
ZG26    Me410
NJG100  Ju88
```

Copyright 1971 The Avalon Hill Co. Software copyright 2015,2016 Brian W. Mulligan.

Single Turn, 20 phases

11 Turns, 20 phases each.
Allies must bomb all cities.

11 Turns, 20 phases each.
Allies must bomb all cities.
Some German units dependant upon r

Build 7 Series 1

780.1796.1034.3849.3330.7.

Apr '45 START GERMAN F 43 I 25 O 16 R 8 C 2 Rpl 10

Plan for this talk

Source control & github

Purpose

Input

Output

Packages / Modules / Libraries

Plan for this talk

Source control & github

Purpose

Input

Output

Packages / Modules / Libraries

Source Control & github

Source Control

Backs up changes to the code over time

Allows tracking changes and the purpose for the changes

Allows reversion to previous versions

Provide your credentials to access this database.

Visual SourceSafe credentials may be different than your Windows network user name and password.

User name: Admin

SourceSafe password:

Database: VSS 8.0

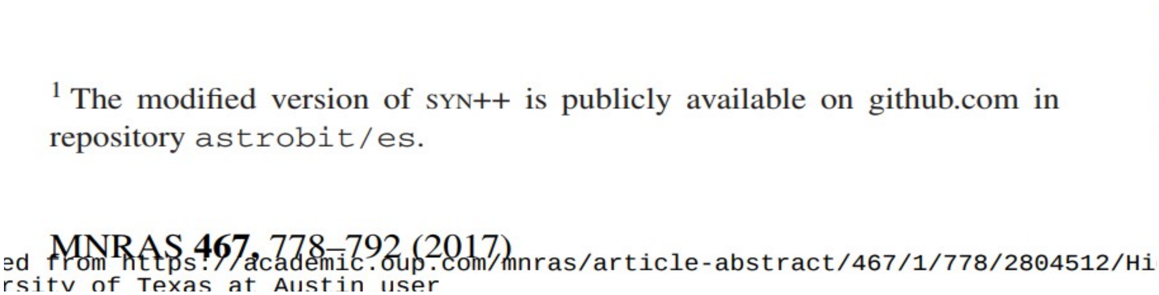
Github: sharing your code

Generally free

As a student, eligible for free private repositories

Share your code with collaborators

Repository for your code used in papers



[1] The modified version of SYN++ is publicly available on github.com in repository astrobit/es.

MNRAS **467**, 778–792 (2017)

ed from https://academic.oup.com/mnras/article-abstract/467/1/778/2804512/Hi
rsity of Texas at Austin user

Plan for this talk

Source control & github

Purpose

Input

Output

Packages / Modules / Libraries

Questions to ask

Does software to do this already exist?

If it does, how complicated is it to make it do what you want to do?

How often do I need to do this task?

Will I need to do it again in the future?

How complex is the task?

How time-consuming is coding, debugging, and processing vs. manual operation?

Graphing

IDL

just don’t

Python – pyplot

Useful for one-off code, a handful of plots, or something that can be done in a short time

C – pgplot

Basis of pyplot, going to run a lot faster than python. Useful for a large number of plots or large datasets

FORTRAN – pgplot

basically the same as C

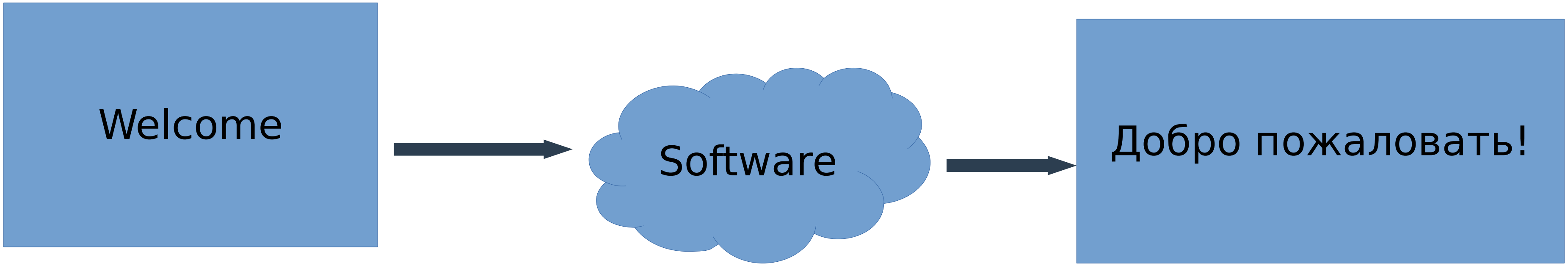
Other

Postscript – skip the intermediary and write your own .ps files; can be done from any language – this is all that pyplot / pgplot does.

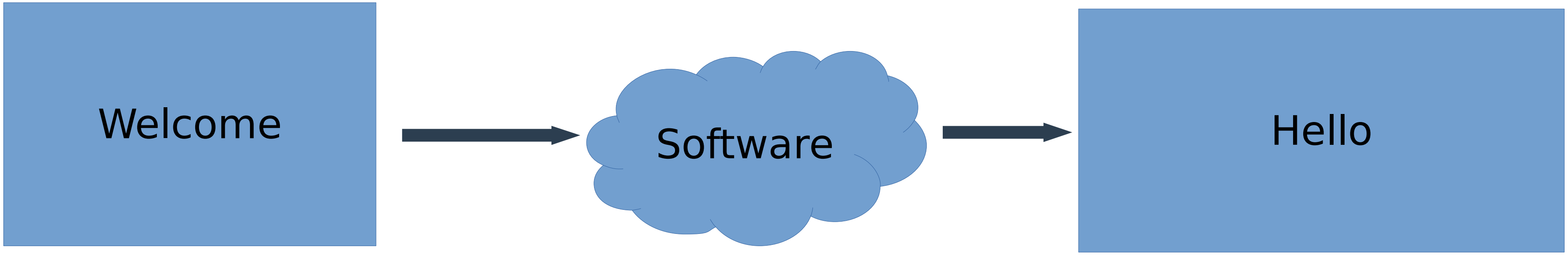
gnuplot, other utilities – probably similar to using python.

Fundamental models

Translation



Manipulation



Purpose / complexity

Generate simple plots or tables?

(Translation model)

Convert data from one format to another?

(Translation model)

Perform calculations on data

(Manipulation model)

Plan for this talk

Source control & github

Purpose

Input

Output

Packages / Modules / Libraries

How to access your data

Assume: data in (set of) files

Avoid hard coding inputs unless they never change

User inputs via:

- command line / typed user input
- graphical user interface
- Parameter files

Command line

Provide parameters to command when running the code

```
g++ -c source.cpp -o source.o –std=c++11
```

Highly flexible and allows user to specify only parameters that are needed.

Requires string processing in code to translate potentially obscure command line options to meaningful information.

Very suitable for scripting

Useful when there is only a handful of parameters

stdin

Provide parameters via stdin

```
imstat
List of input images (HIP62157_firstpointing-0001_V.fit):
```

Useful for code that is run a few times manually

Can make use of redirecting stdin so all parameters can be stored in one or more files.

Using stdin means unformatted parameter files – not particularly user friendly

Parameter files

Store parameter information in a formatted file

Parameter file with fixed filename (don’t do this!) or via command line

Formats:

- xml: broadly used standard, lots of code available for processing
- json: starting to come into broad use; features similar to xml, but not as verbose
- yaml: similar to json or xml, but a little more limited; not as widely used
- Parameter list:

  ```
  parameter_a = 1
  parameter_b = whats up?
  ```

  Requires more intensive string processing in your code, but pretty easy to read

Data input

What storage format is the data in?

What format do you need it in your code?

These two questions should guide you to appropriate data structures and libraries / packages for reading the data.

Do you need all of it all the time?

For large datasets, you'll probably only want to load what you need when you need it.

Reading from disk is really slow.

How much memory / space is required for the data?

What order will you access the data?

For large datasets, be aware of access order – you want to make sure that access is ordered the same as storage.

Plan for this talk

Source control & github

Purpose

Input

Output

Packages / Modules / Libraries

Final products

Human readable?

Machine readable?

Both?

Graphs?

LaTeX tables?

Machine Readable files

Xml, json, .csv, others

Maintain precision of floating point data?

- Single precision: 8 digits
- Double precision: 17 digits

Binary machine readable?

- Most compact format
- Can cause problems if data is transferred from one computer to another

Intermediate products?

Storage of intermediate state of processed data

- Useful if processing is intensive
- Allows restart in case of crash
- Usually machine readable format
- Format is highly dependent on usage
- Consider SQL

Other intermediate data

- Information regarding the analysis results
- Effectively an output product

Plan for this talk

Source control & github

Purpose

Input

Output

Packages / Modules / Libraries

Modules & Libraries

What components of your code do you expect to re-use?

Do you find yourself regularly re-using a piece of code with small changes?

Can those components be written in a way to be usable in different contexts?

Functional or object oriented programming

Helps prevent errors and allows fixing bugs in a global manner.

Find bug in a piece of code, if it's in a library, fix it once, otherwise fix it 10 times.

Interface design

What parameters are required?

What parameters are optional?

Many modern languages allow optional paramters to have a default value

```
void thisfunction(int a, int &b, int c = 10)
```

Consider grouping parameters into a container

Highly flexible method of supplying paramters

```
class params
{
  public:
  int a;
  int b;
  int c;
  params(void) {c = 10;}
};
void thisfunction (params Params);
```

Github!

Place your modules / libraries / packages on github – other people can now use them!

Remember to provide some documentation to users so they know how to use the code (and you remember when you come back to it a year later)

Github.com/astrobit/xlibs

Summary

Consider usage and scope of task

Set up inputs to be easy to control, avoid hard-coding (fixed) inputs

Outputs as needed – consider language

Create libraries / modules to avoid errors and copy / paste

Use github and share your work!