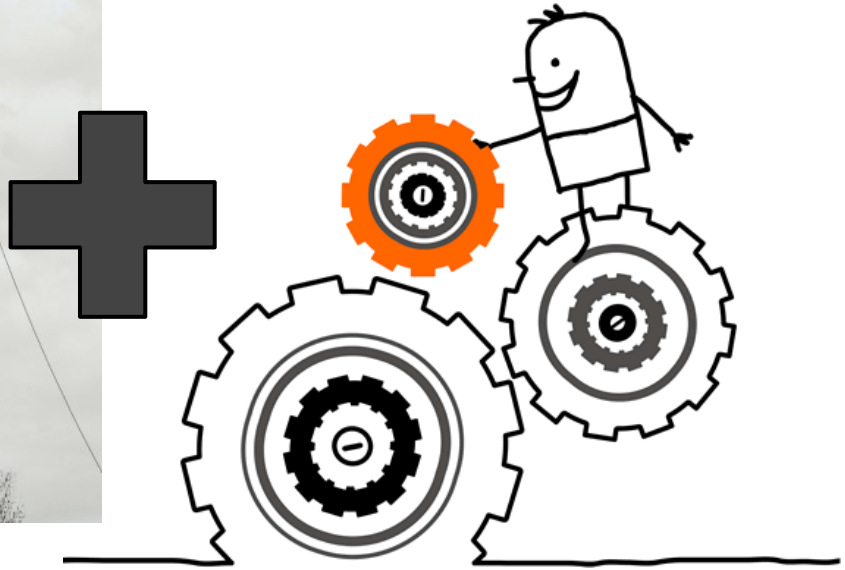
Asset Build Systems

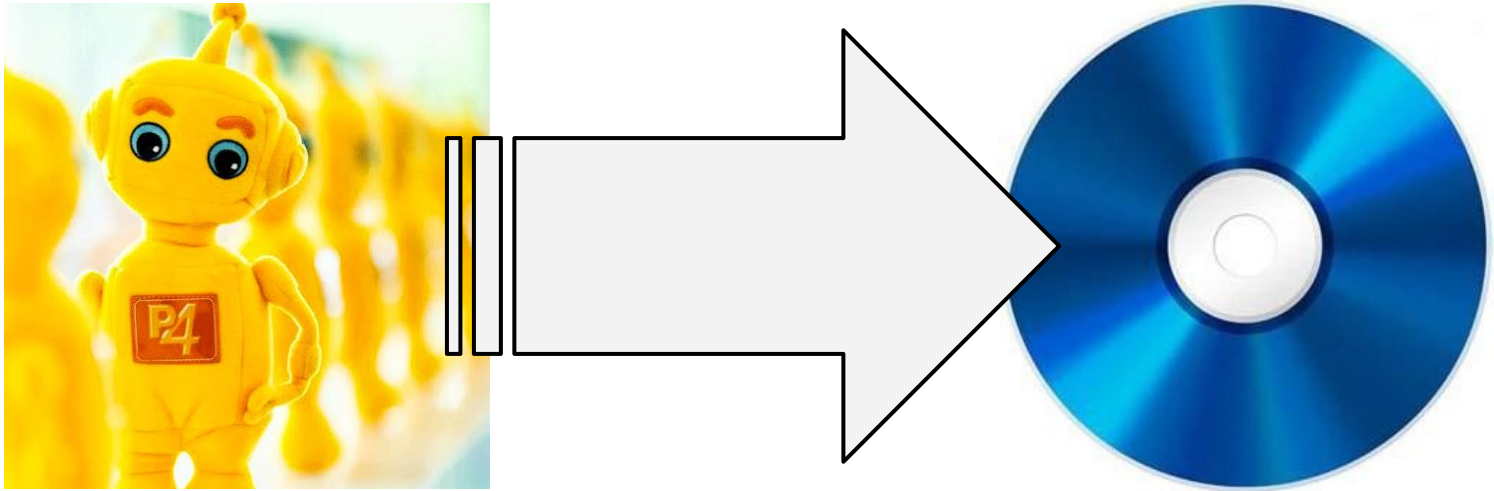
Kris Lang
Client Technical Director
Game Technology Group @ SOE

GAME DEVELOPERS CONFERENCE®
SAN FRANCISCO, CA
MARCH 17–21, 2014
EXPO DATES: MARCH 19–21
2014

Asset Build Systems



What is an asset build system?




```
C:\>Perforce2DiscImage.exe
```

We don't need an Asset build System

A modern tale of horror

THE FOLLOWING **PREVIEW** HAS BEEN APPROVED FOR

ALL AUDIENCES

BY THE MOTION PICTURE ASSOCIATION OF AMERICA, INC.

THE FILM ADVERTISED HAS BEEN RATED

R

RESTRICTED

UNDER 17 REQUIRES ACCOMPANYING PARENT OR ADULT GUARDIAN

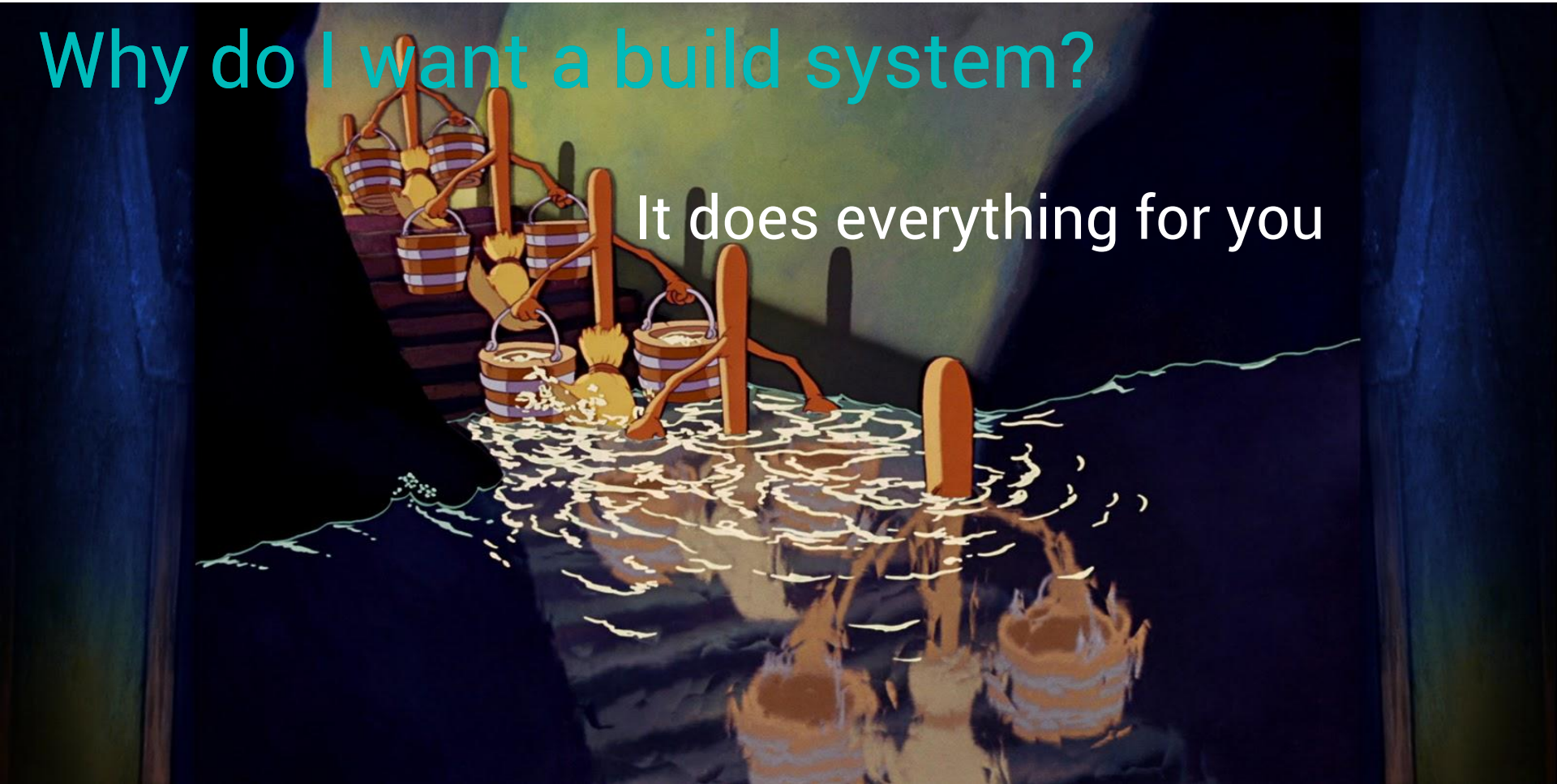
Asset Build Systems

Kris Lang
Client Technical Director
Game Technology Group @ SOE

GAME DEVELOPERS CONFERENCE®
SAN FRANCISCO, CA
MARCH 17–21, 2014
EXPO DATES: MARCH 19–21
2014

Why do I want a build system?

Why do I want a build system?



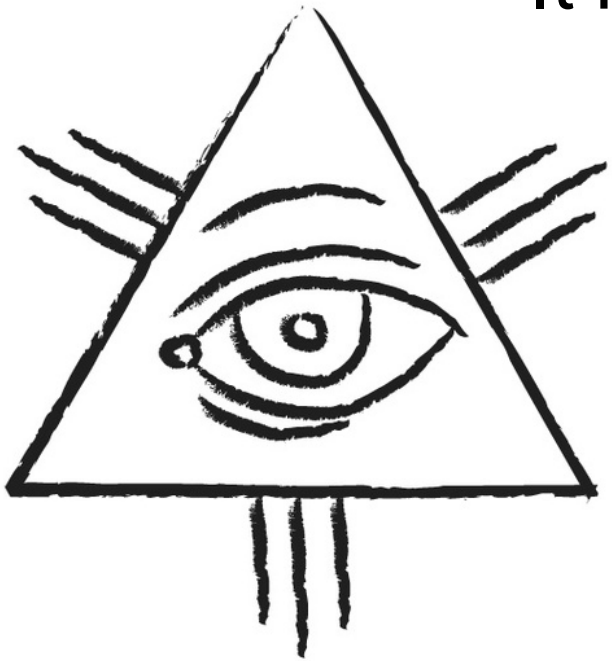
It does everything for you

Why do I want a build system?

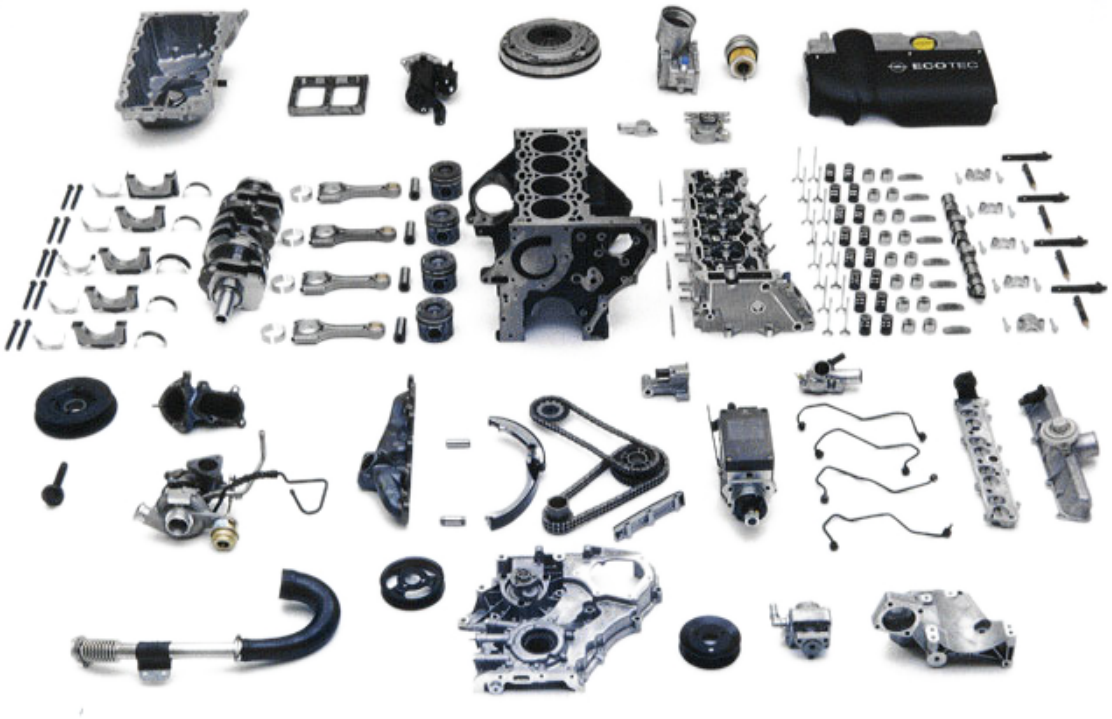
Everyone can run the entire process

Why do I want a build system?

It knows about everything

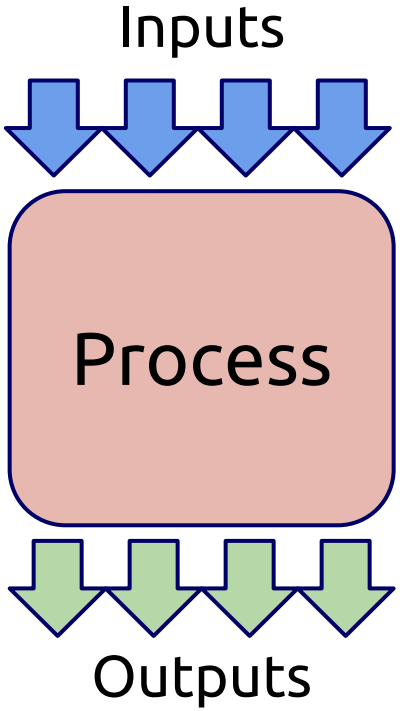


What are the parts of a build system?



What are the parts of a build system?

List inputs



What are the parts of a build system?

List inputs

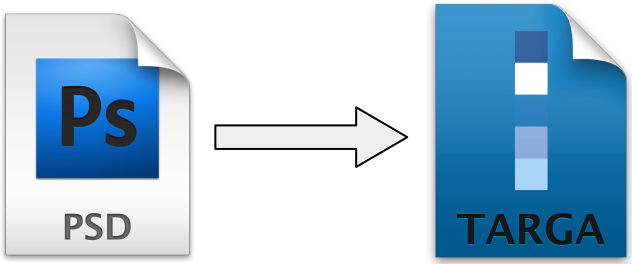
Determine settings

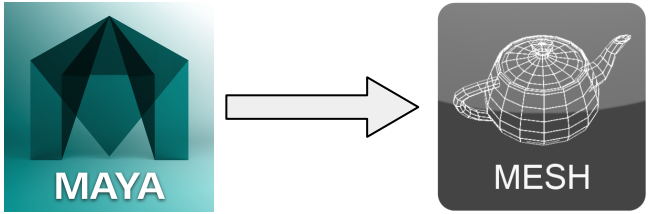
What are the parts of a build system?

List inputs

Determine settings

Data conversion tools





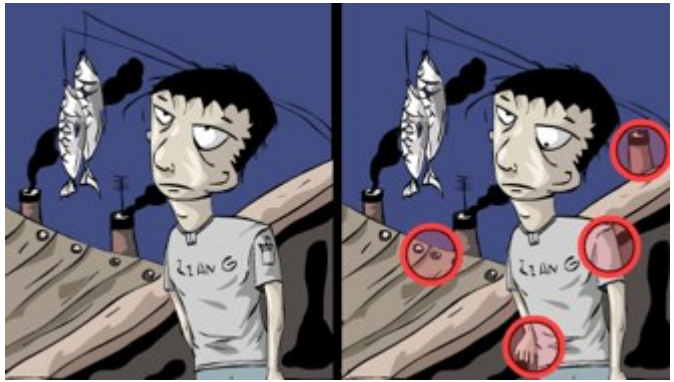
What are the parts of a build system?

List inputs

Determine settings

Data conversion tool

Change detection



What are the parts of a build system?

List inputs

Determine settings

Data conversion tool

Change detection

Failure handling

What are the parts of a build system?

List inputs

Determine settings

Data conversion tool

Change detection

Failure handling

</what is an asset build system>

The rest of this talk assumes you think this is the best idea ever

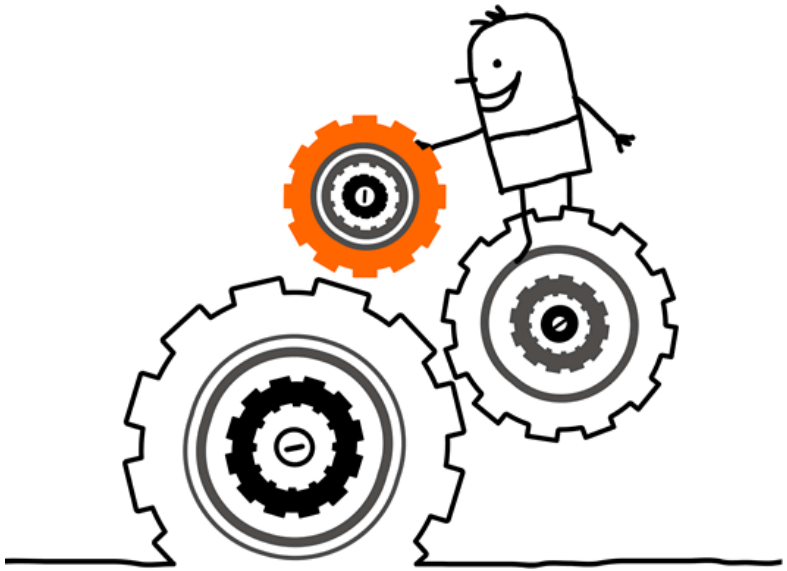
- intermission -

This sounds awfully programmery,
why should TA be involved? ಠ_ಠ

How does a build system work?

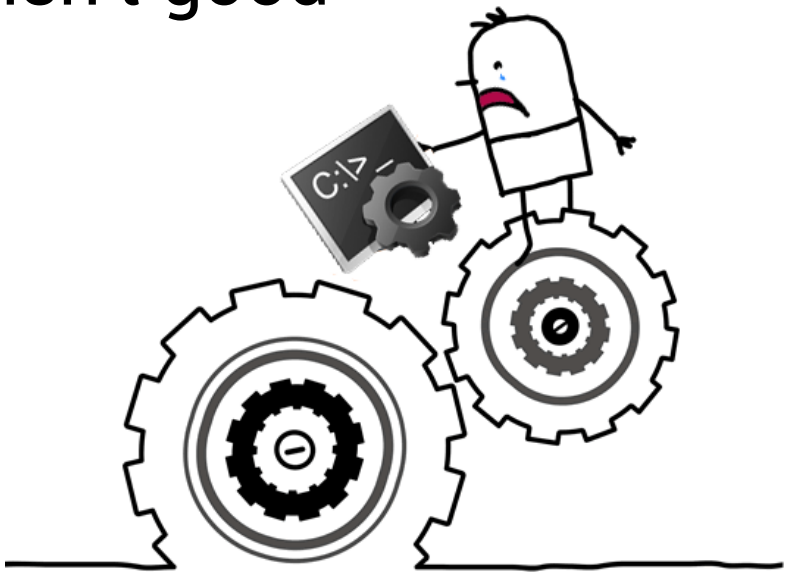
I'm sold on the idea, what do I need to know?

What makes it a build *system*?



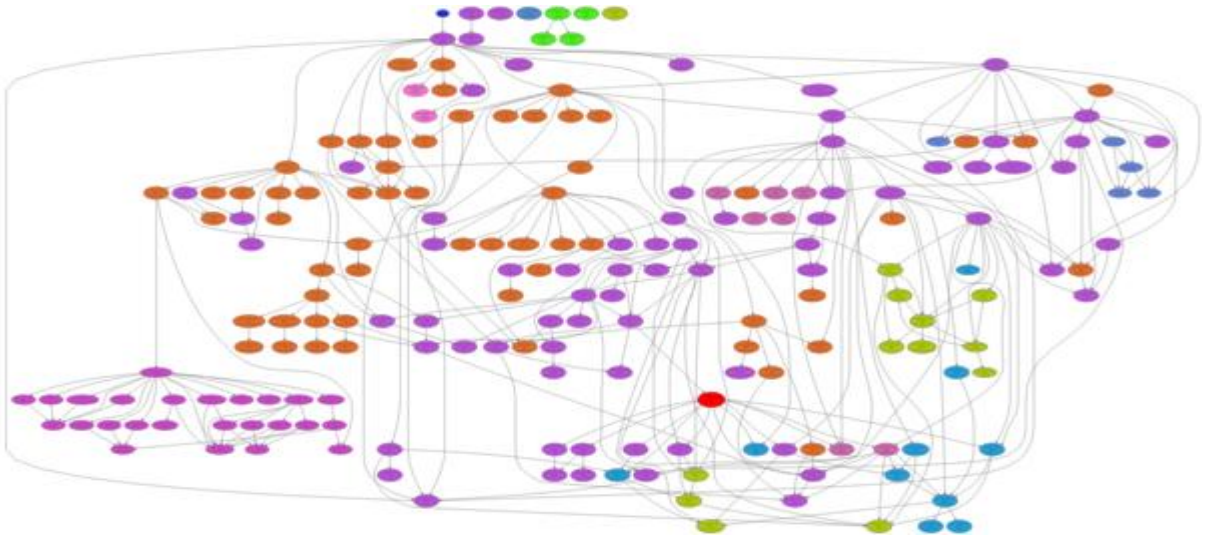
What makes it a build *system*?

a.k.a. Why a batch file isn't good enough



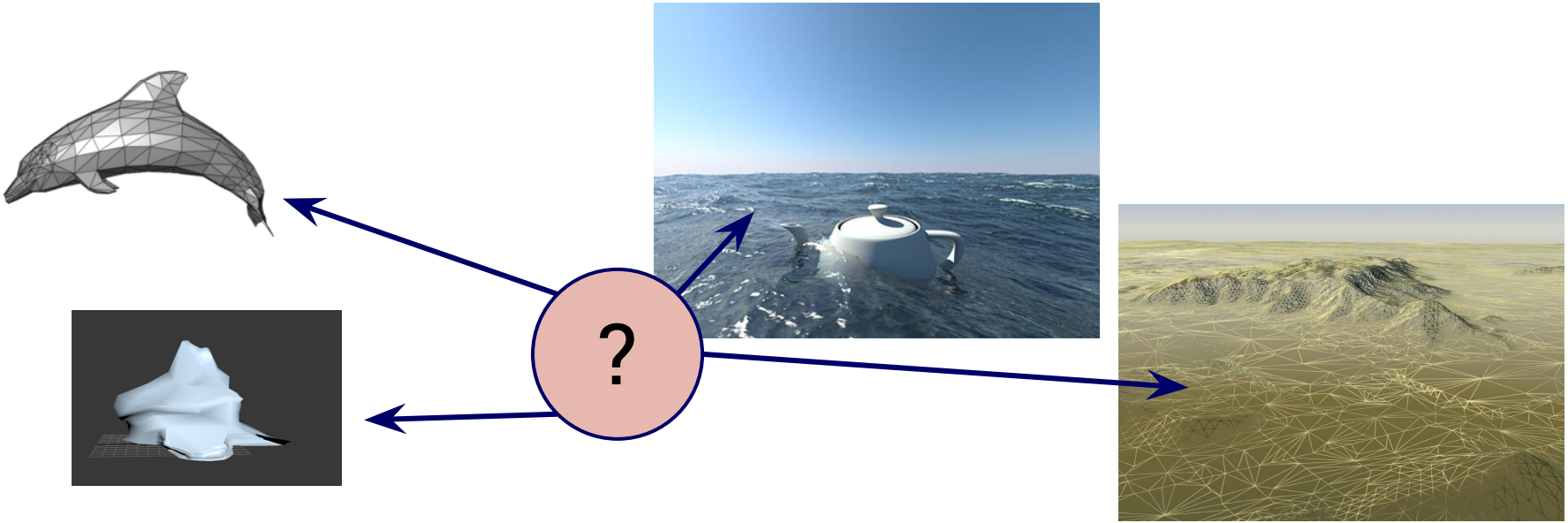
What makes it a build *system*?

It does dependency checking and change detection



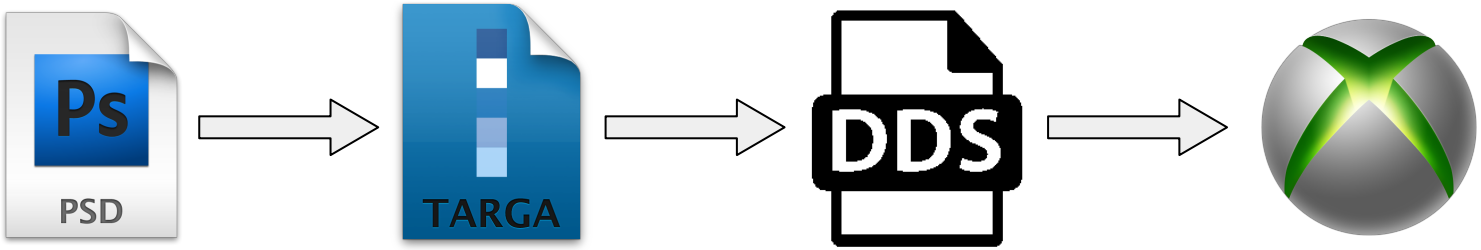
What makes it a build *system*?

It considers more than one asset at a time



What makes it a build *system*?

It does multiple levels of data conversion



What makes it a build *system*?

… it does dependency checking and change detection

… it considers more than one asset at a time

… it does multiple levels of data conversion

Do I have to make one from scratch?

… it does dependency checking and change detection

… it considers more than one asset at a time

… it does multiple levels of data conversion

That feels like a lot of work...

Do I have to make one from scratch?

… it does … king and change …

… it cons … e asset at a time

… it does … data convers …

A selection of build tools

A selection of build tools

Batch files

Well, it'll work

A selection of build tools

make

The granddaddy of all build systems

```
 * so old
there's no
   icon
```

A selection of build tools

msbuild

xml based, command line processing

A selection of build tools

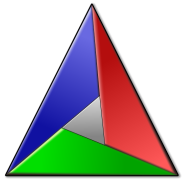
ant

xml based, extensible via java

A selection of build tools

cmake

custom language, generates native builds



A selection of build tools

scons

end to end python

How do we use these tools?

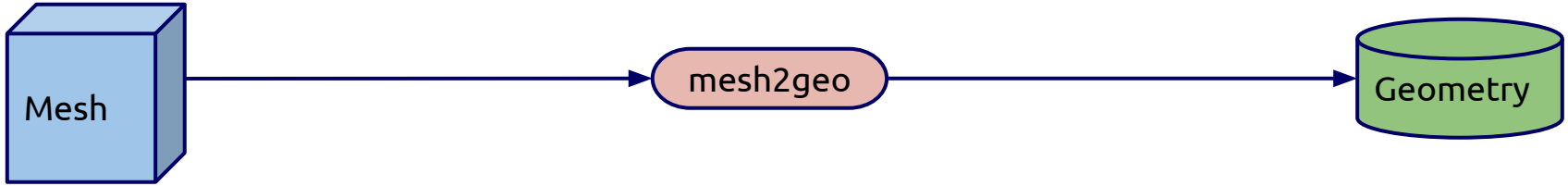
How do we use these tools?

S P T

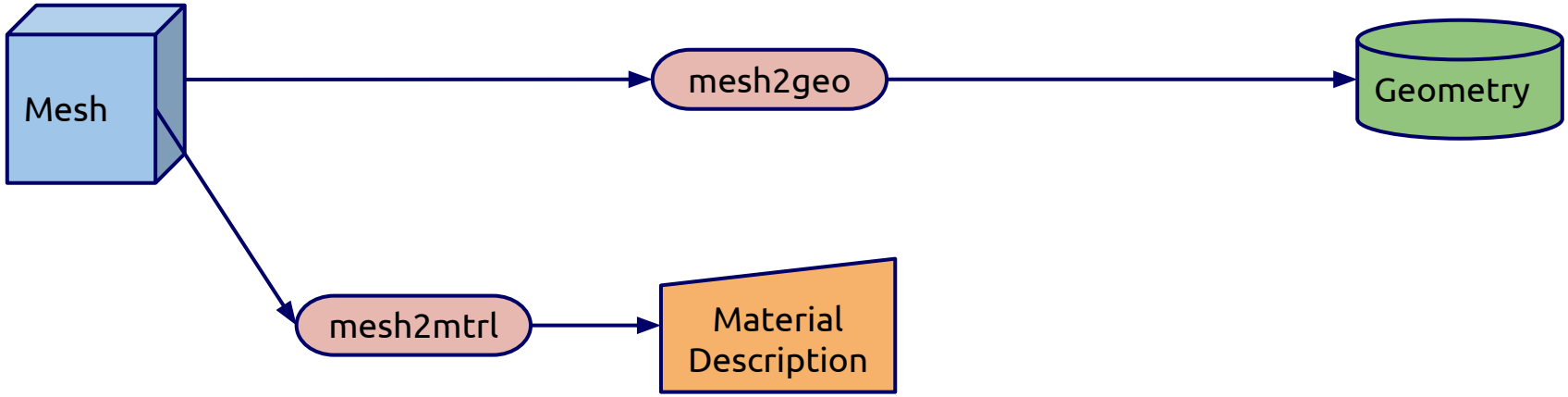
sources → processing → targets

?

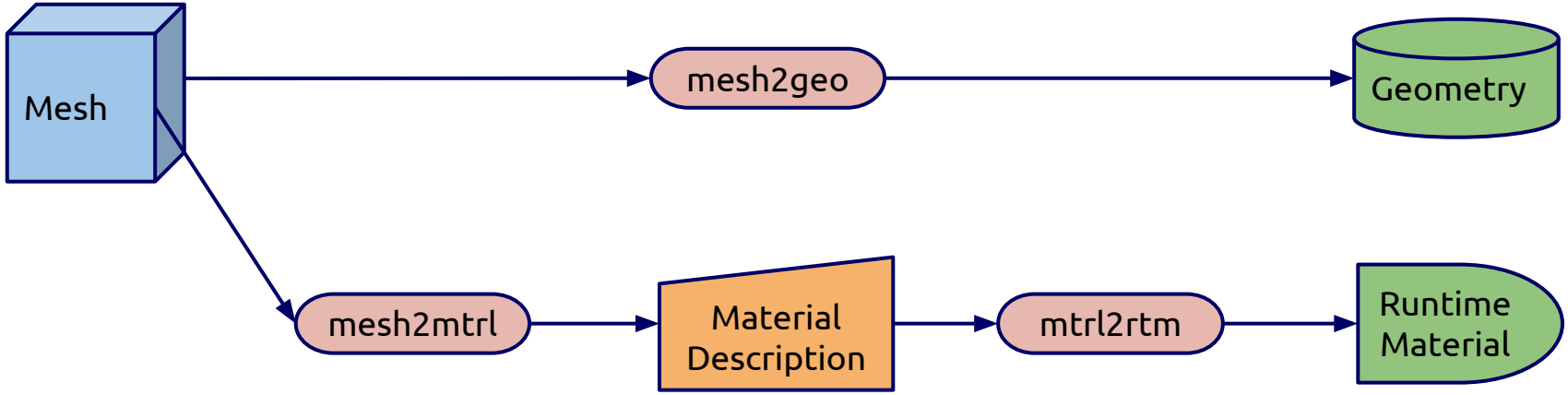
SPT



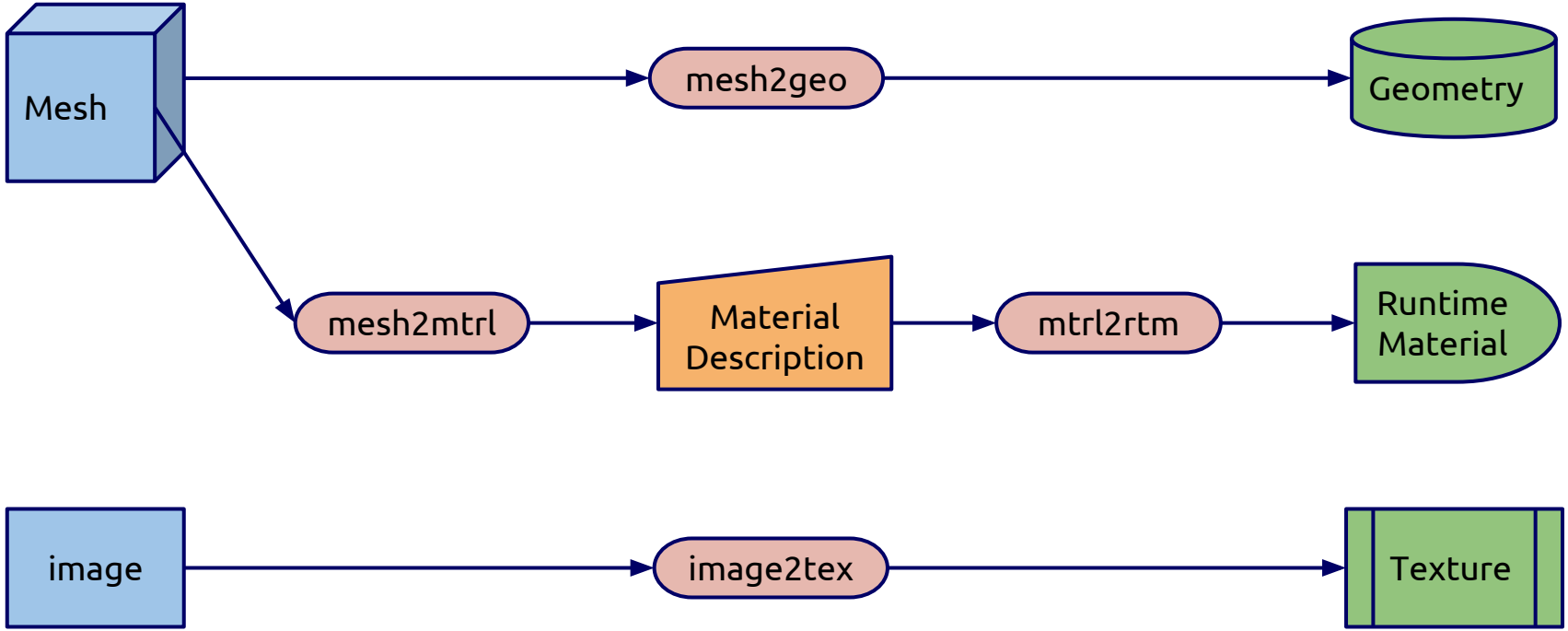
SPT



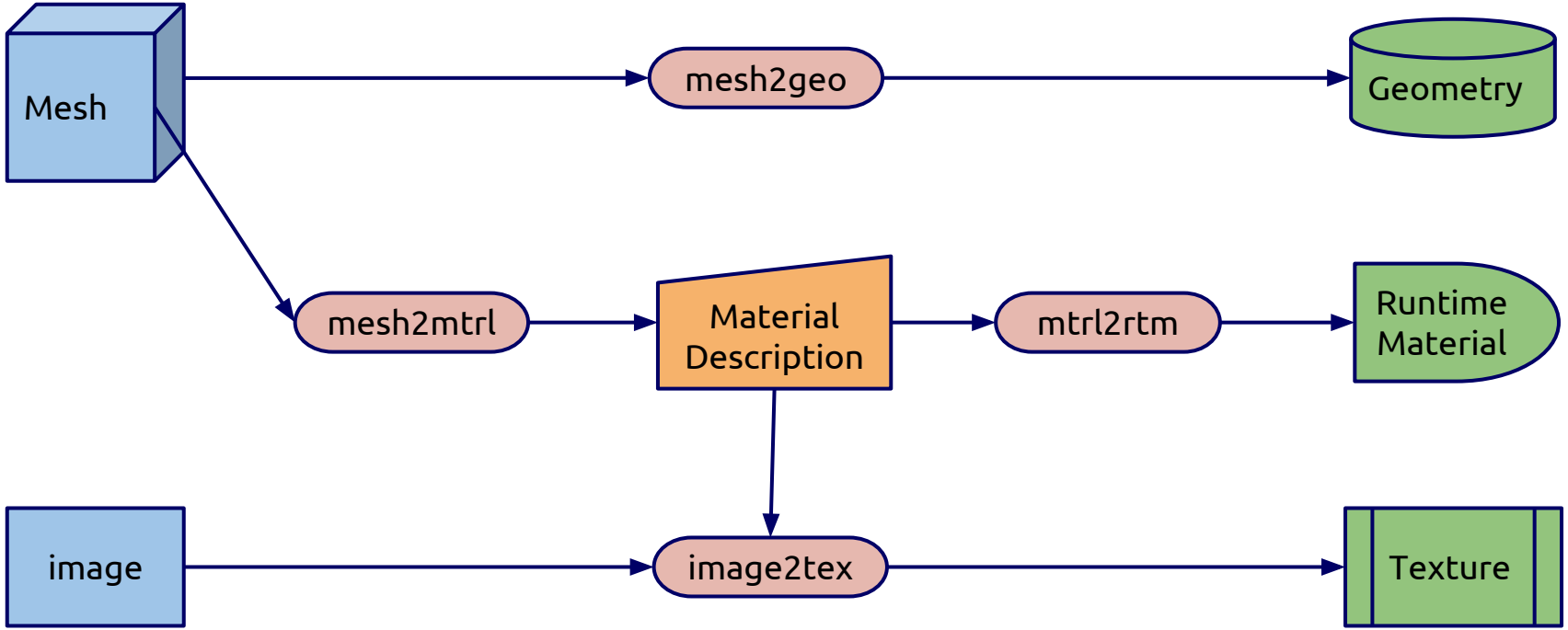
SPT


Mesh
mesh2geo
Geometry
mesh2mtrl
Material Description
mtrl2rtm
Runtime Material

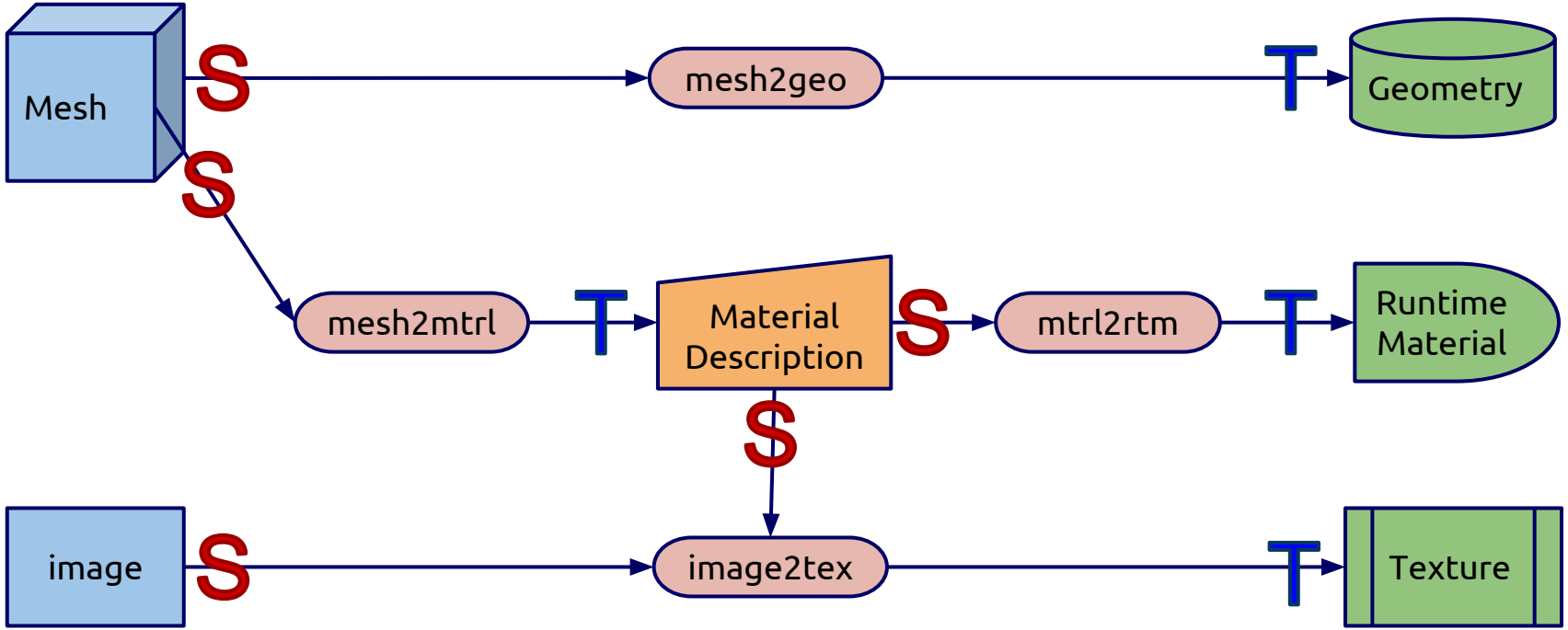
SPT


Mesh
mesh2geo
Geometry
mesh2mtrl
Material Description
mtrl2rtm
Runtime Material
image
image2tex
Texture

SPT


Mesh
mesh2geo
Geometry
mesh2mtrl
Material Description
mtrl2rtm
Runtime Material
image
image2tex
Texture

SPT


Mesh
S
mesh2geo
T
Geometry
S
mesh2mtrl
T
Material Description
S
mtrl2rtm
T
Runtime Material
S
image
S
image2tex
T
Texture

Build system SPT

<psuedo_code>

Build system SPT

Describe the processing for every type of file that you have

```
for typeOfFile in ourGame:

    processor = Processor(typeOfFile.whatToDo)
    build.processors[typeOfFile] = processor
```

Build system SPT

Tell it about instances of those files

```
for sourceFile in ourGame:

    processor = build.processors[sourceFile.type]
    target = processor.add(sourceFile)
    targets += target
```

Build system SPT

Tell it about the dependent files…
repeat

```
while( len(targets) > 0 ):

    for sourceFile in targets:
        processor = build.processors[sourceFile.type]
        newTargets += processor.add(sourceFile)

    targets = newTargets
```

Example

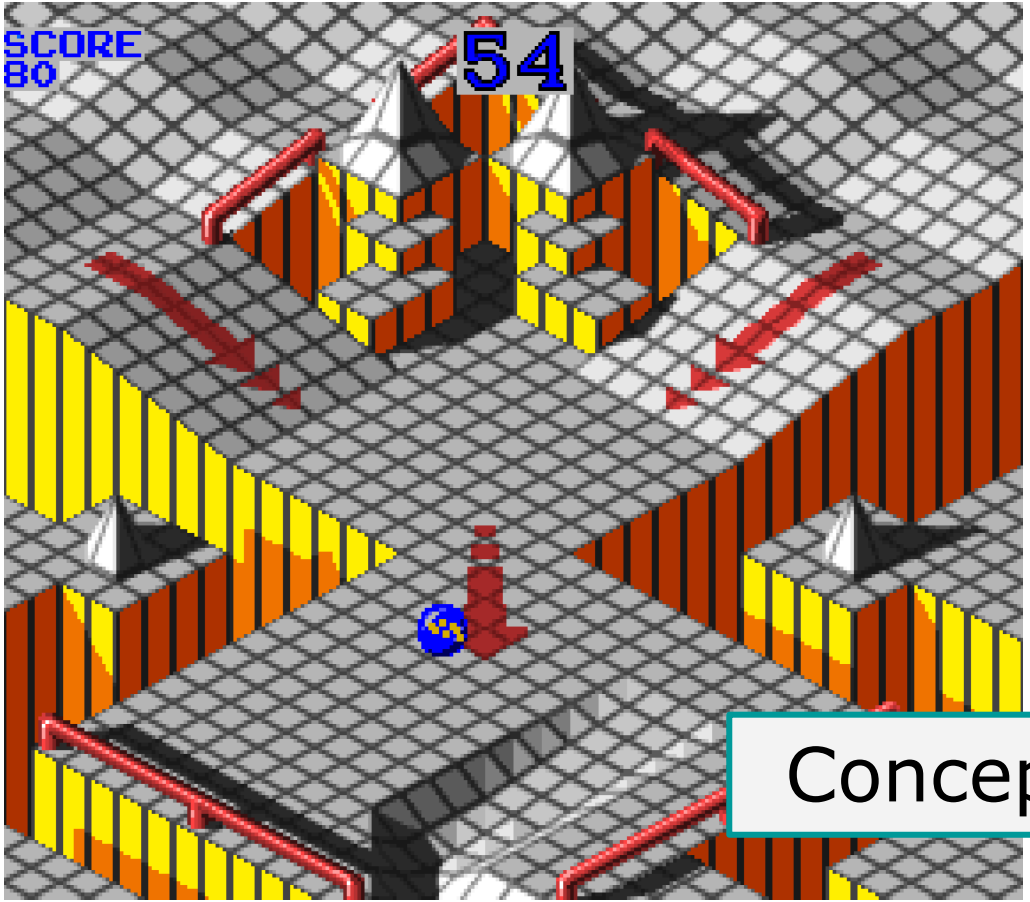
</psuedo_code>

Example

Brilliant game pitch…

- There’s a ball rolling around a level, and you can’t let it fall off

Example



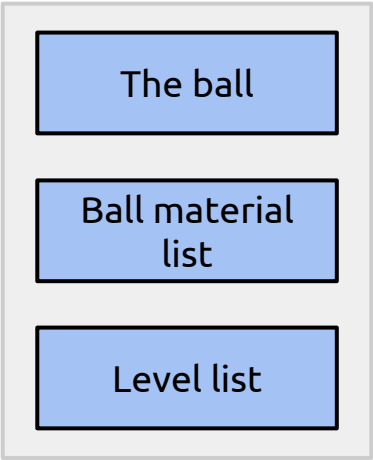
Concept only

Example

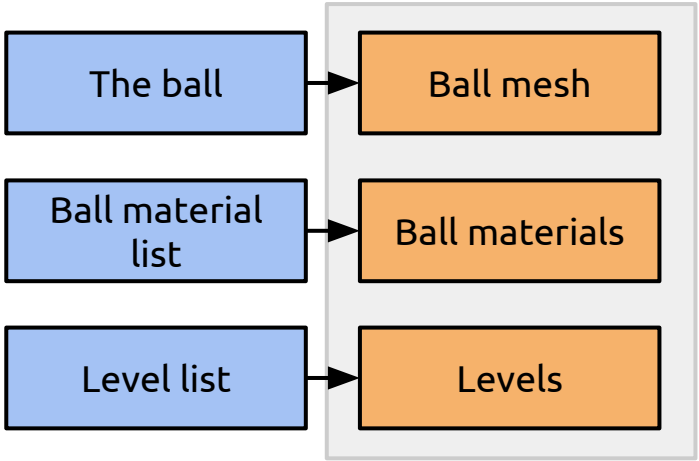
Asset Details:

- Multiple ball materials, one mesh
- Multiple Levels
- Levels have props on them

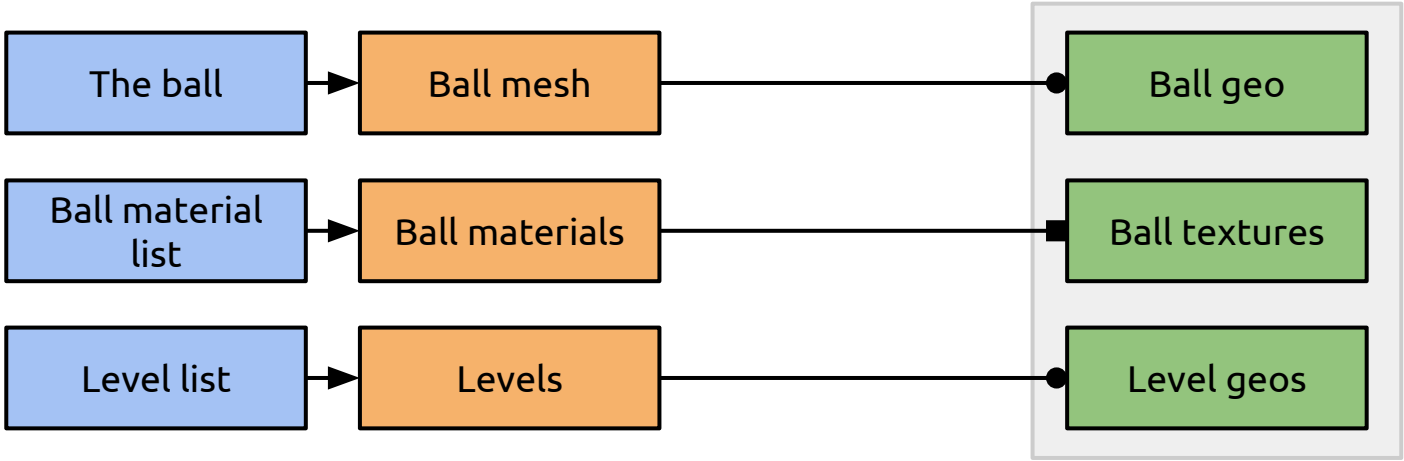
Example



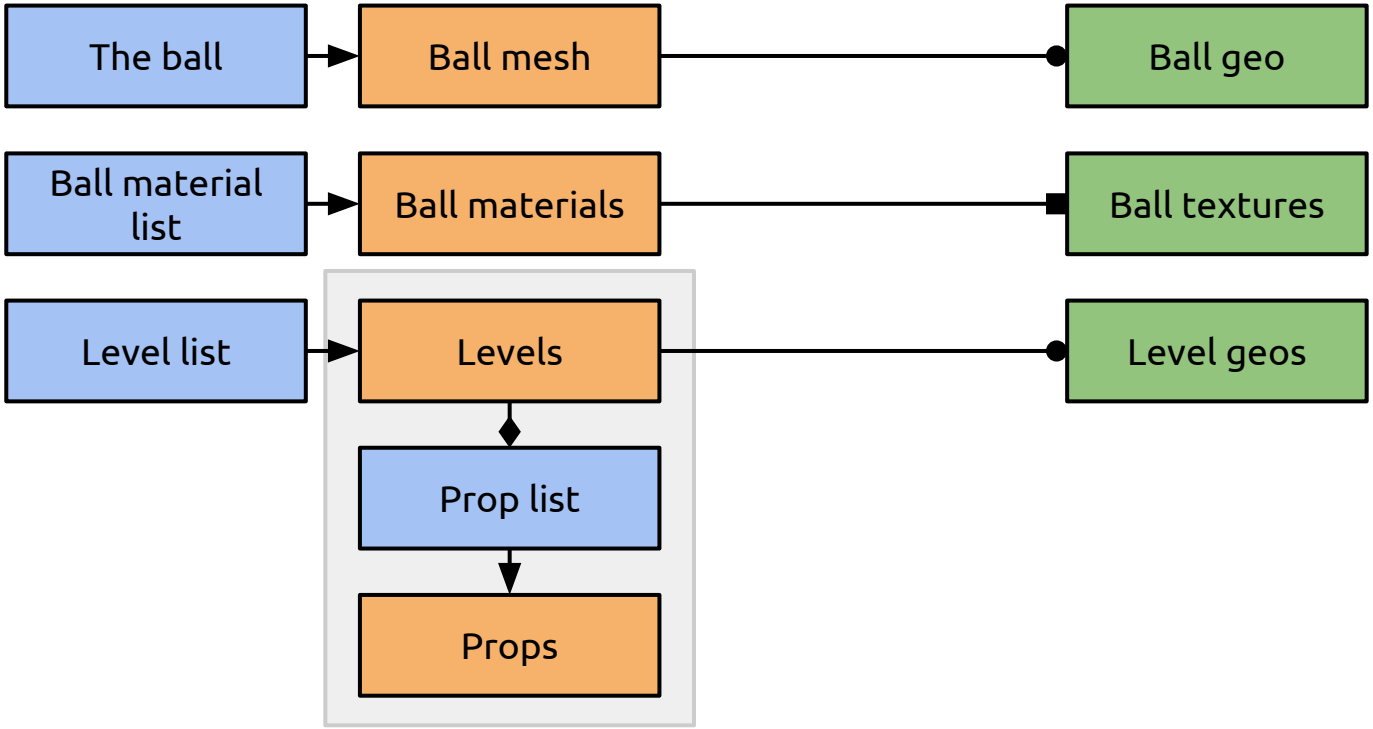
Example


The ball
Ball mesh
Ball material list
Ball materials
Level list
Levels

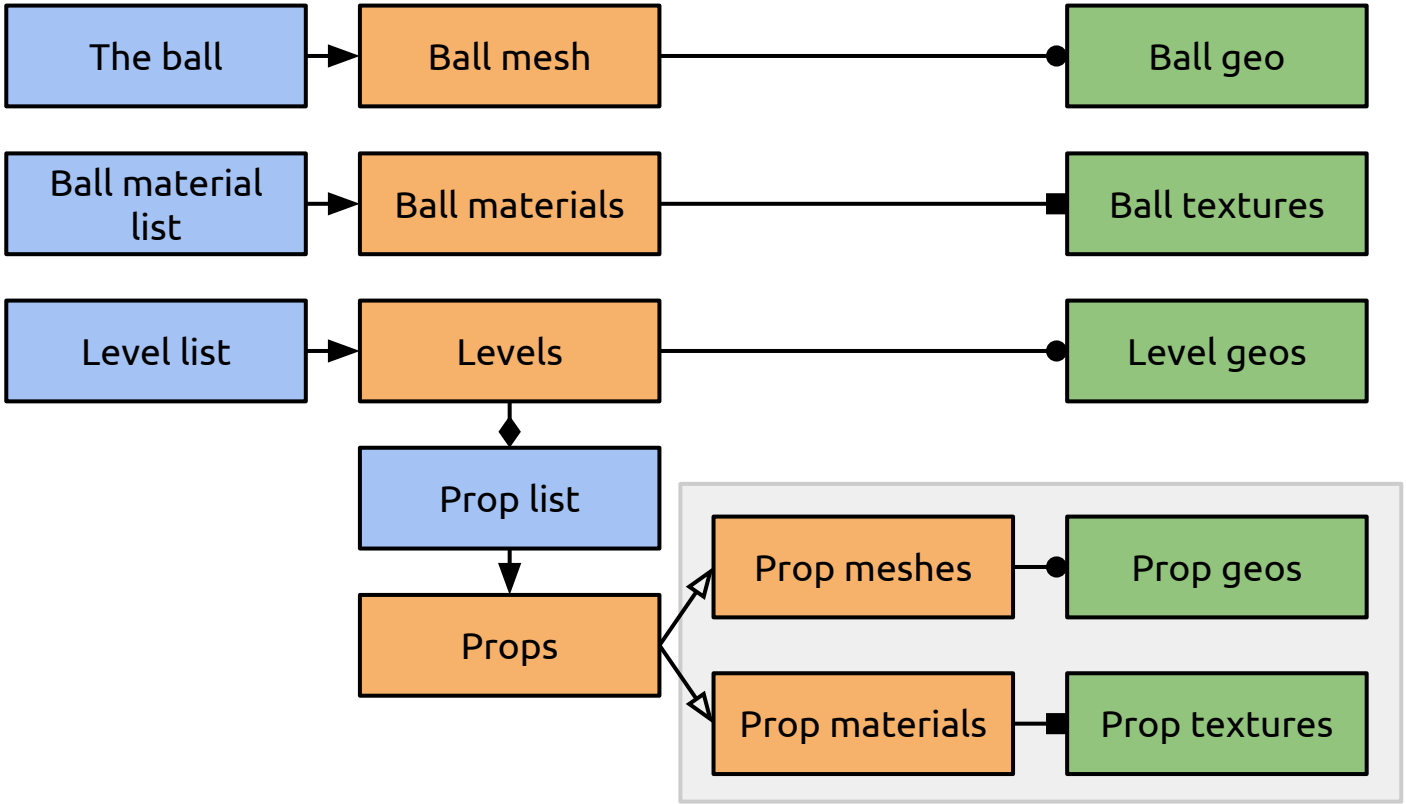
Example


The ball
Ball mesh
Ball geo
Ball material list
Ball materials
Ball textures
Level list
Levels
Level geos

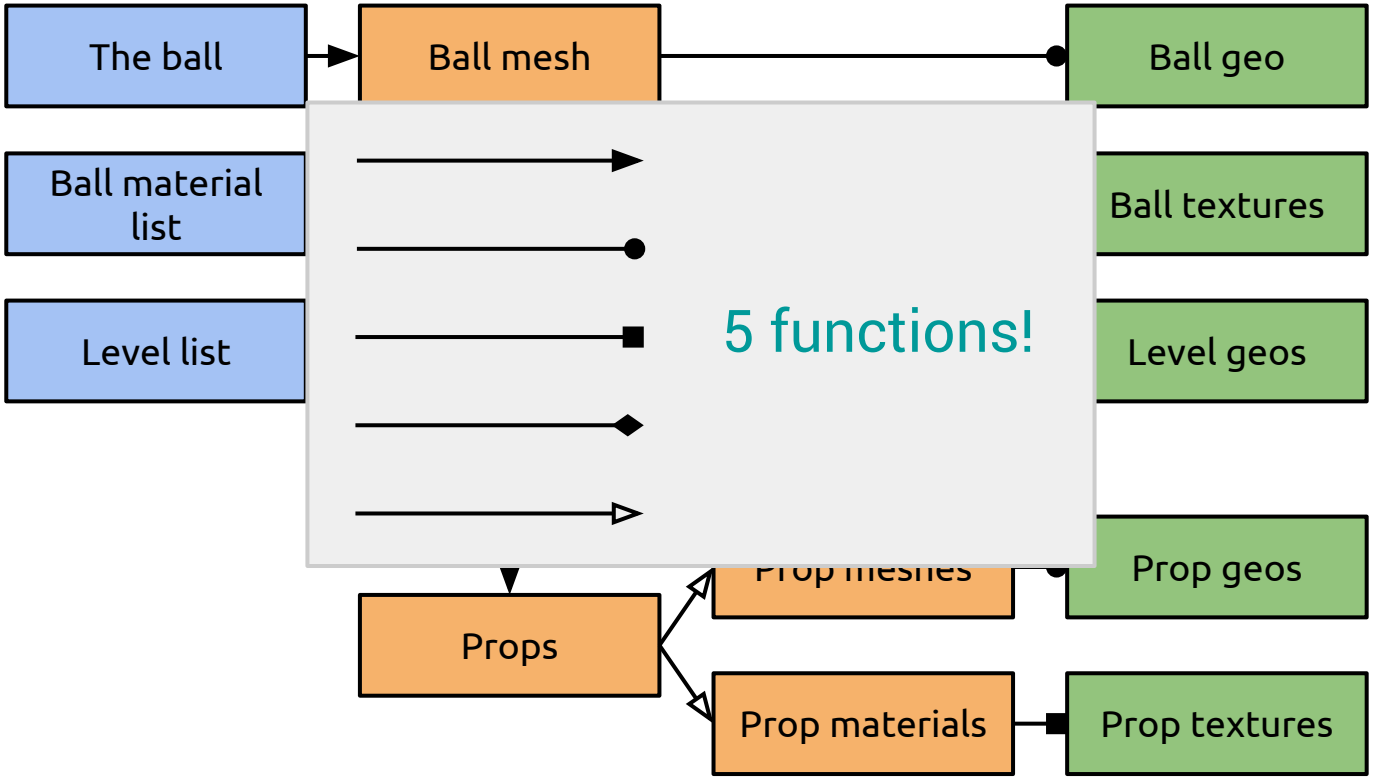
Example


The ball
Ball mesh
Ball geo
Ball material list
Ball materials
Ball textures
Level list
Levels
Level geos
Prop list
Props

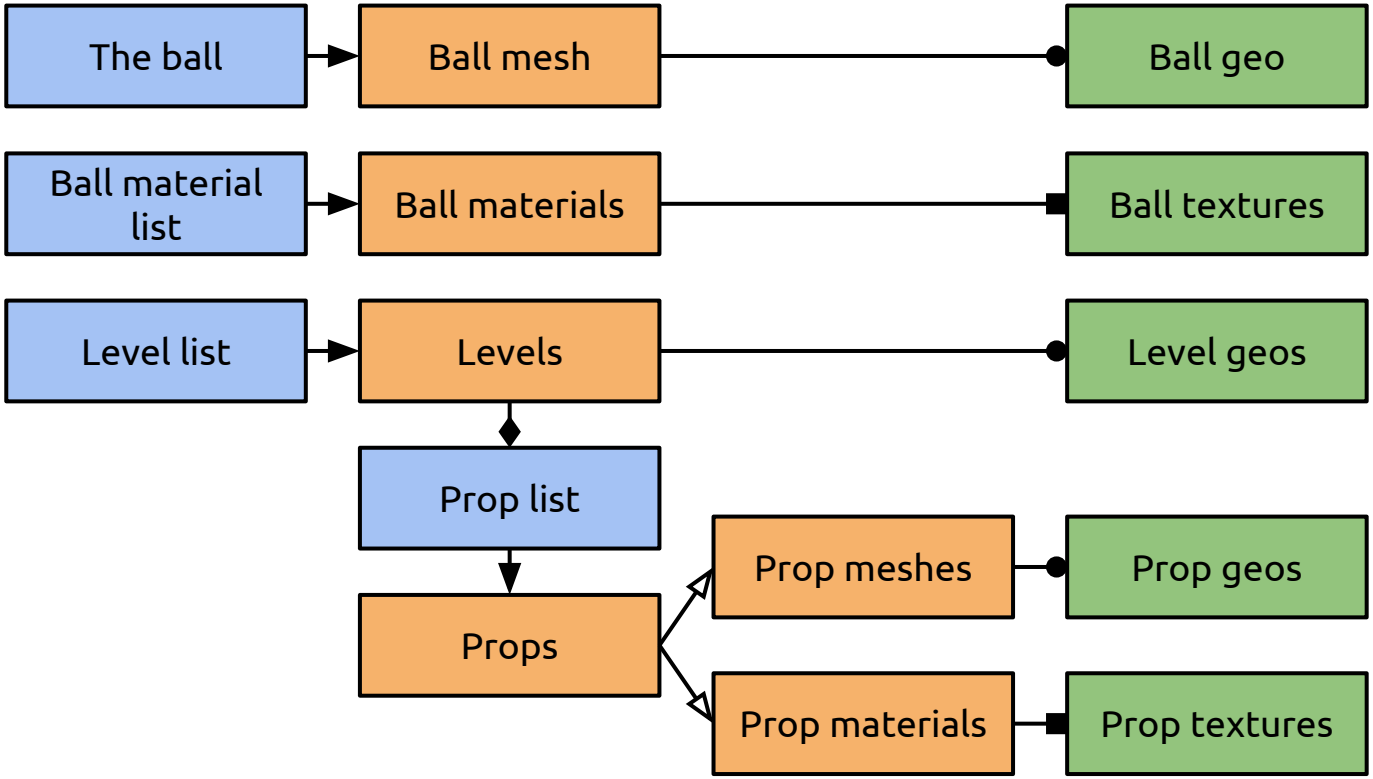
Example


The ball
Ball mesh
Ball geo
Ball material list
Ball materials
Ball textures
Level list
Levels
Level geos
Prop list
Prop meshes
Prop geos
Props
Prop materials
Prop textures

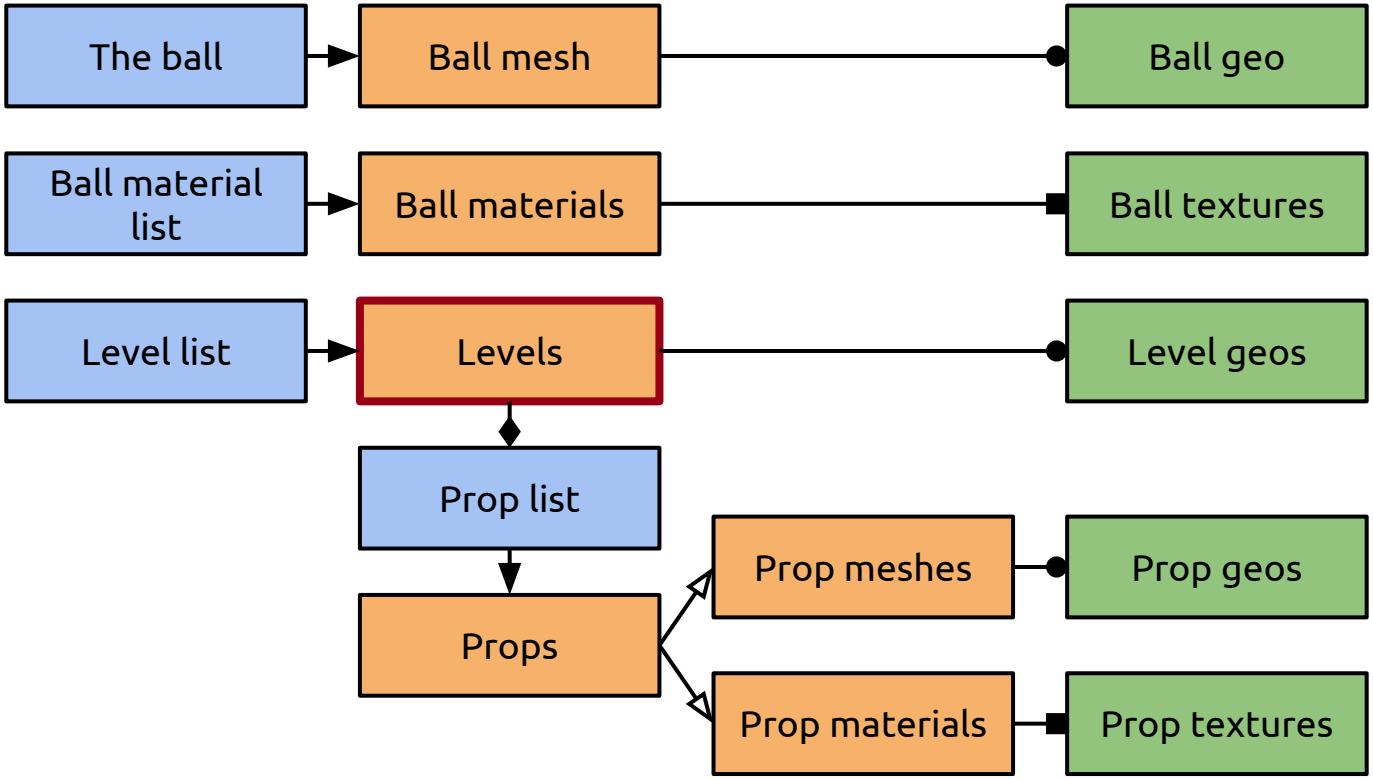
Example


The ball
Ball mesh
Ball geo
Ball material list
Ball textures
Level list
5 functions!
Level geos
Prop geos
Prop meshes
Props
Prop materials
Prop textures

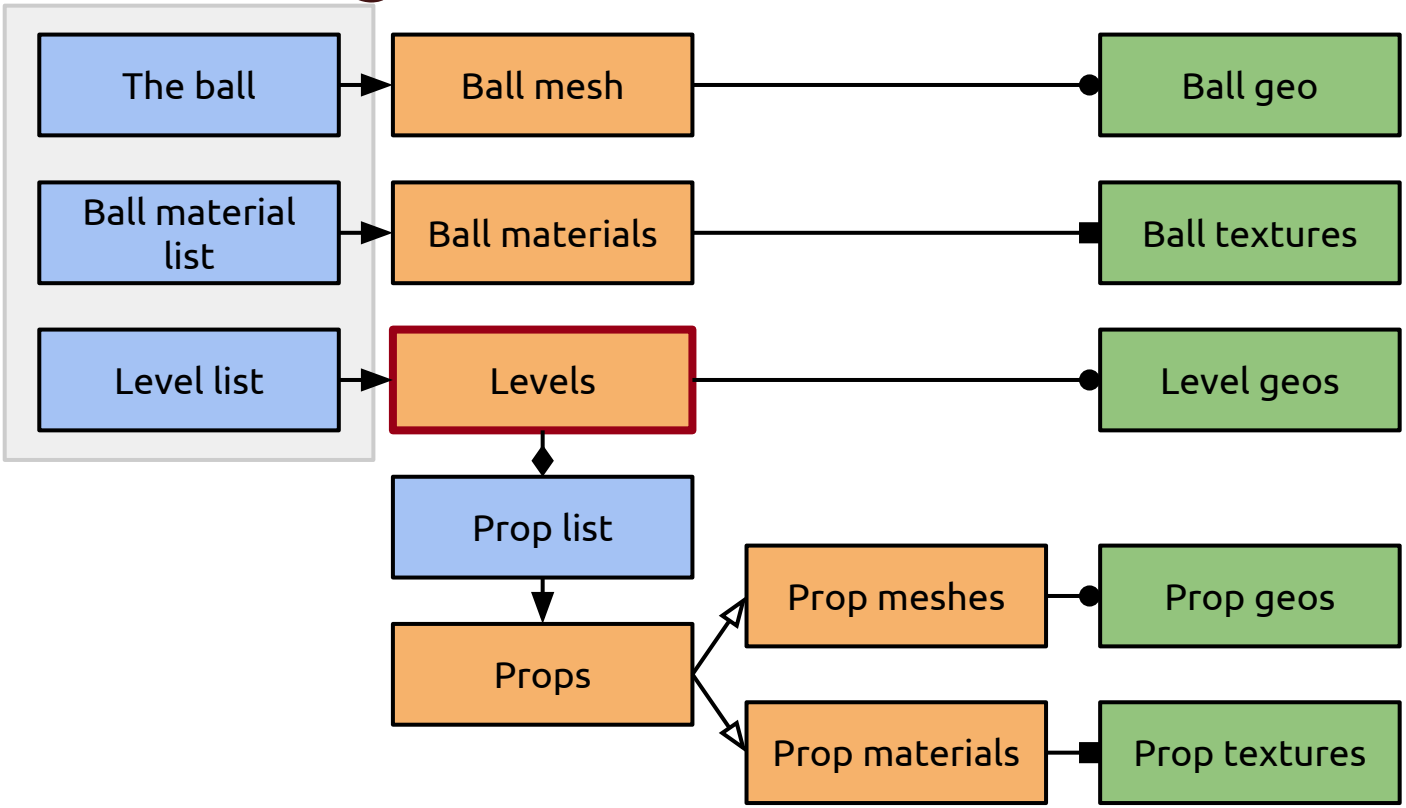
After a change



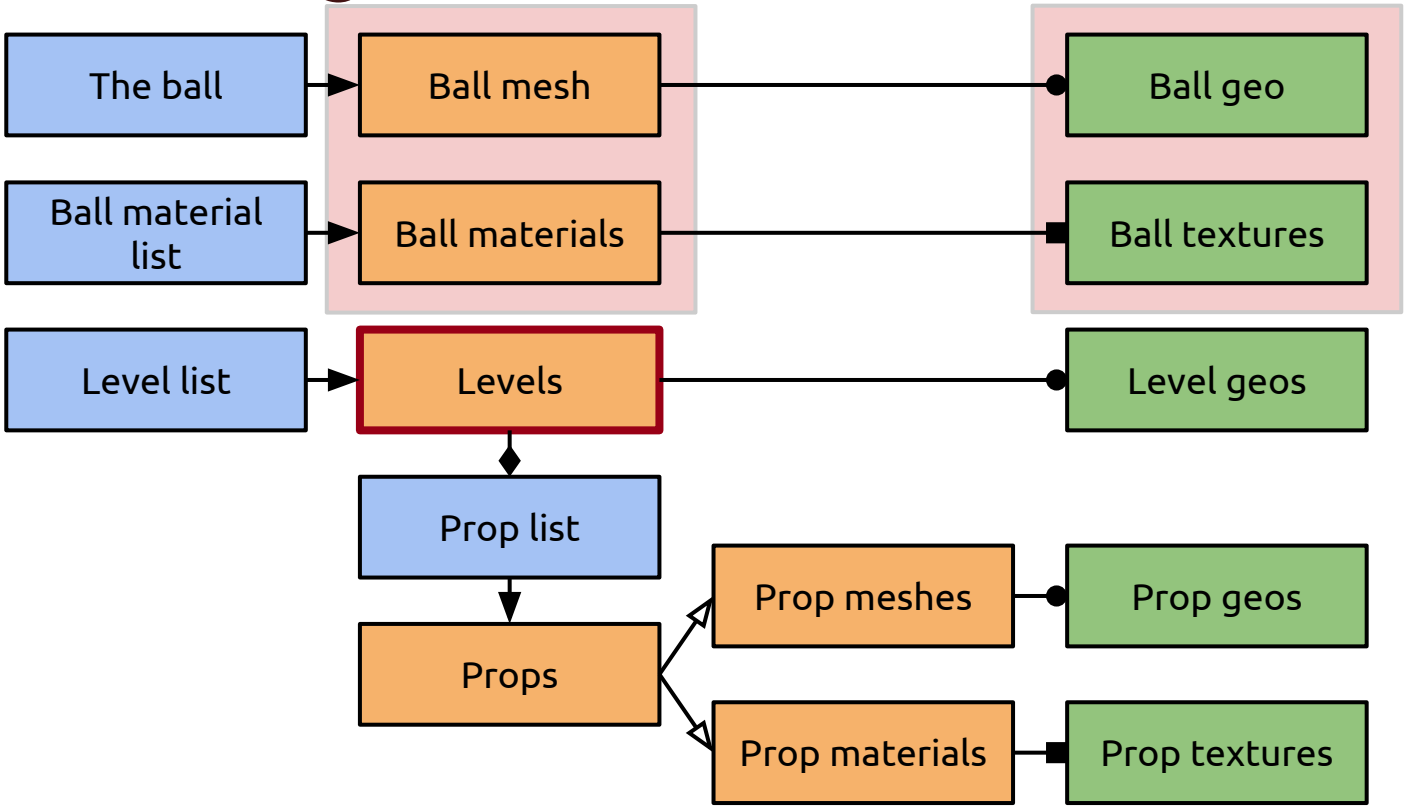
After a change



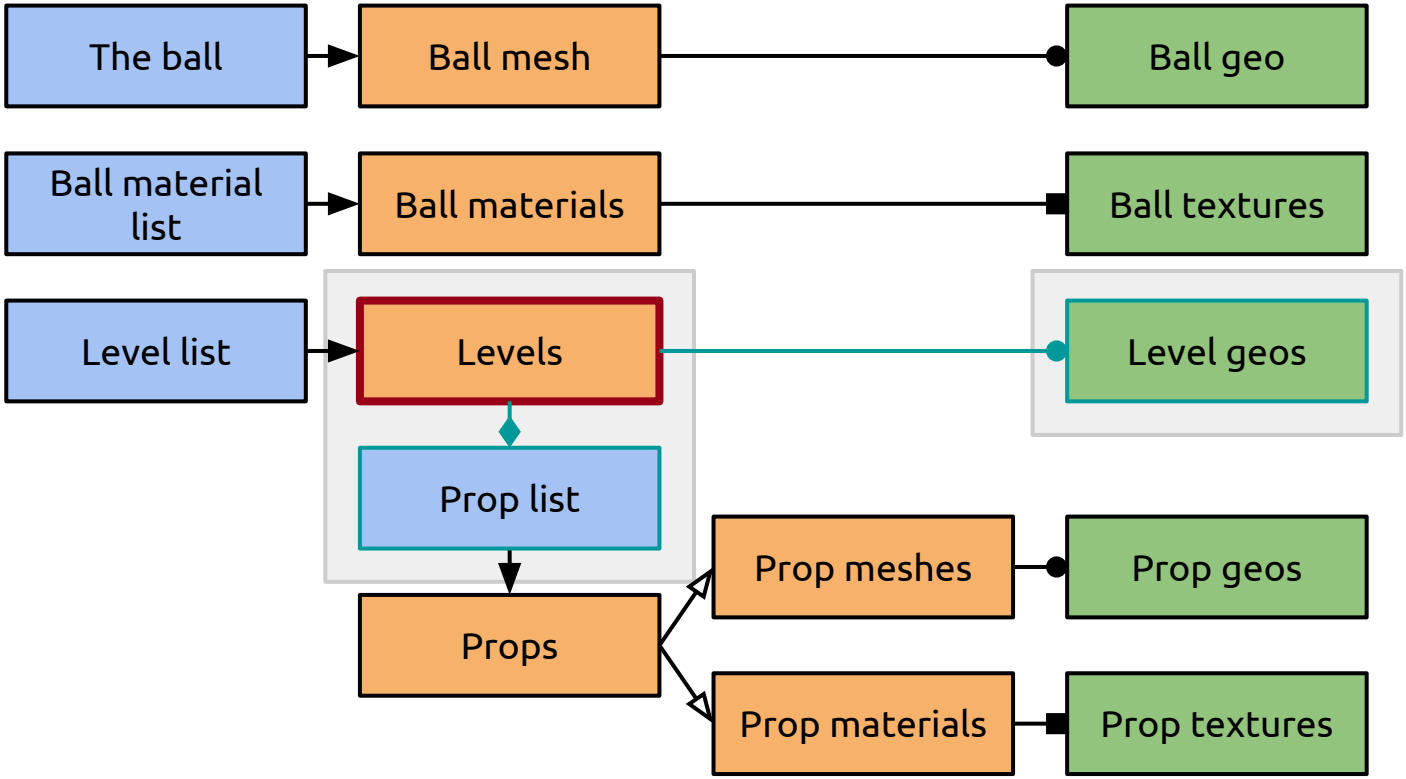
After a change



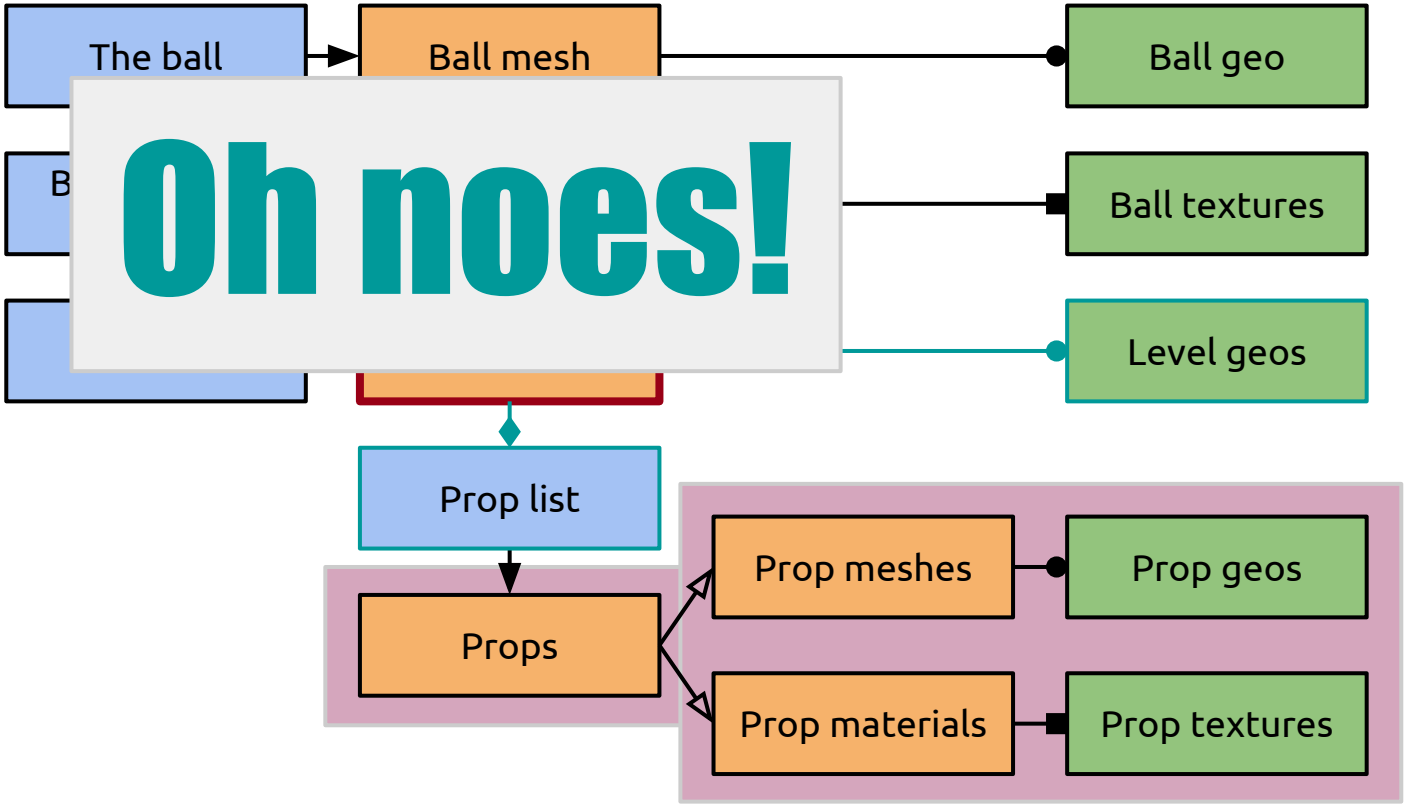
After a change



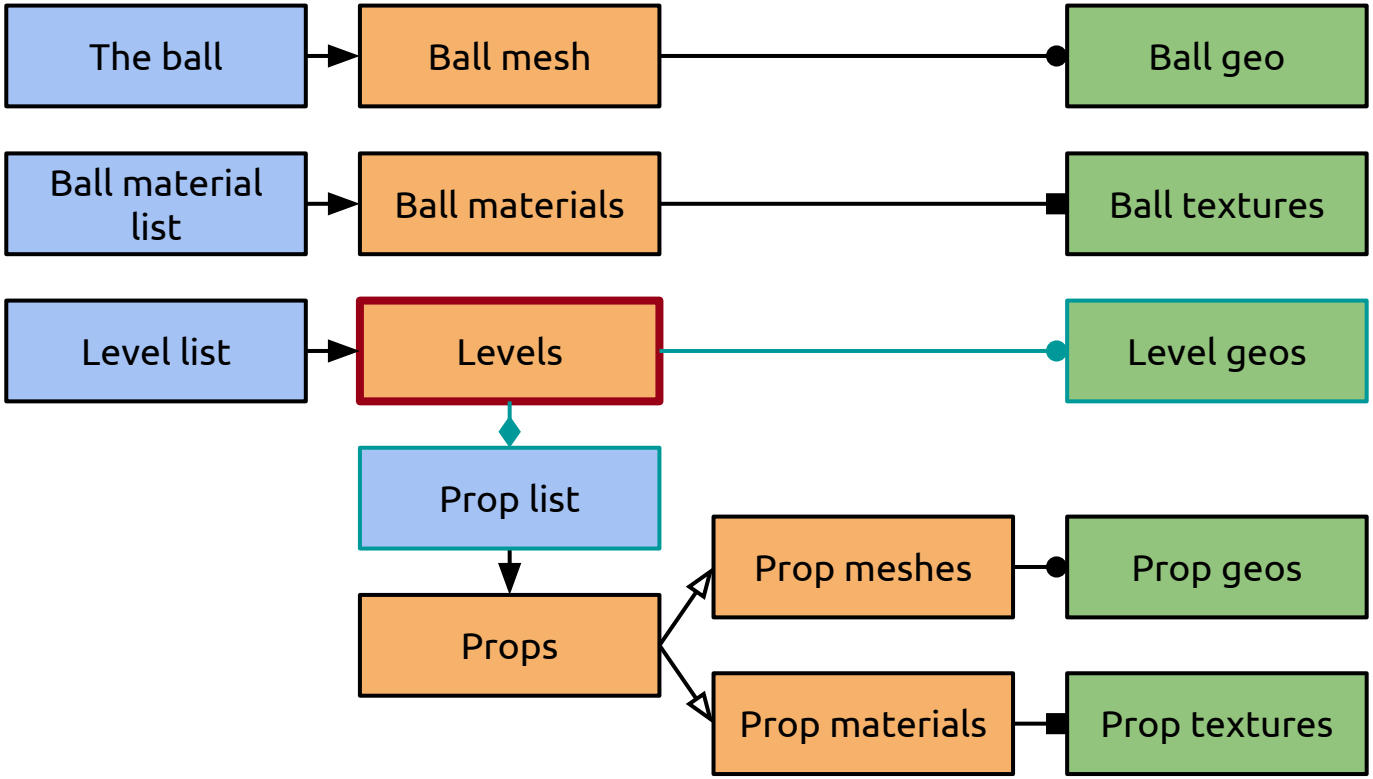
After a change



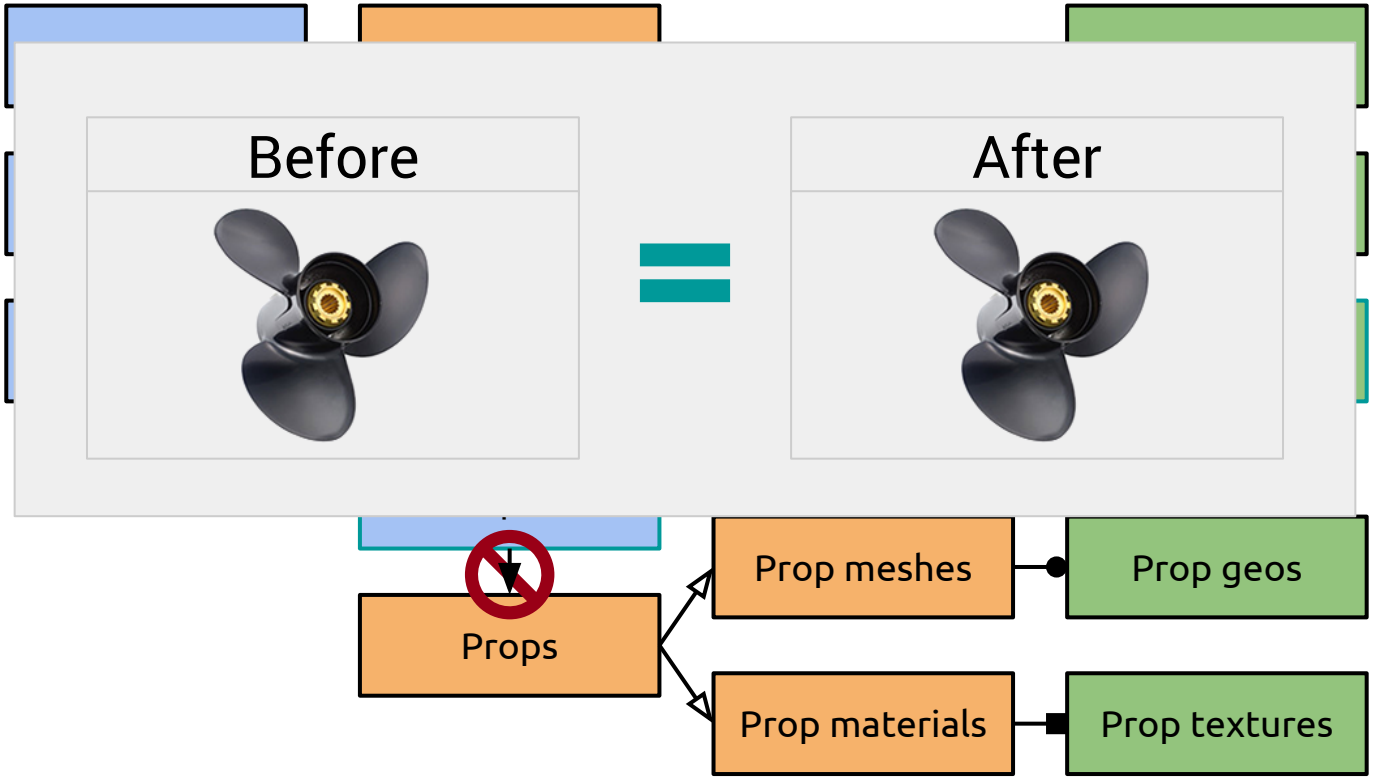
After a change



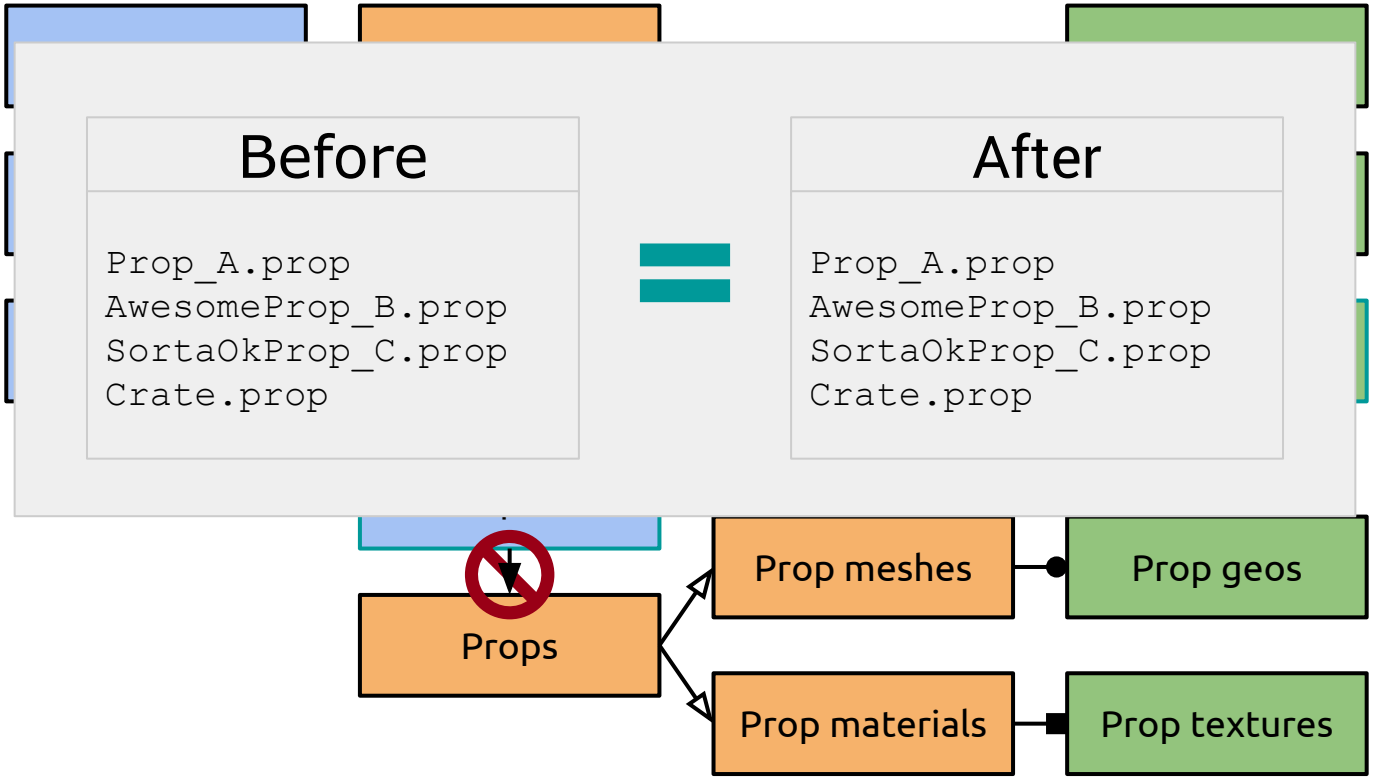
After a change



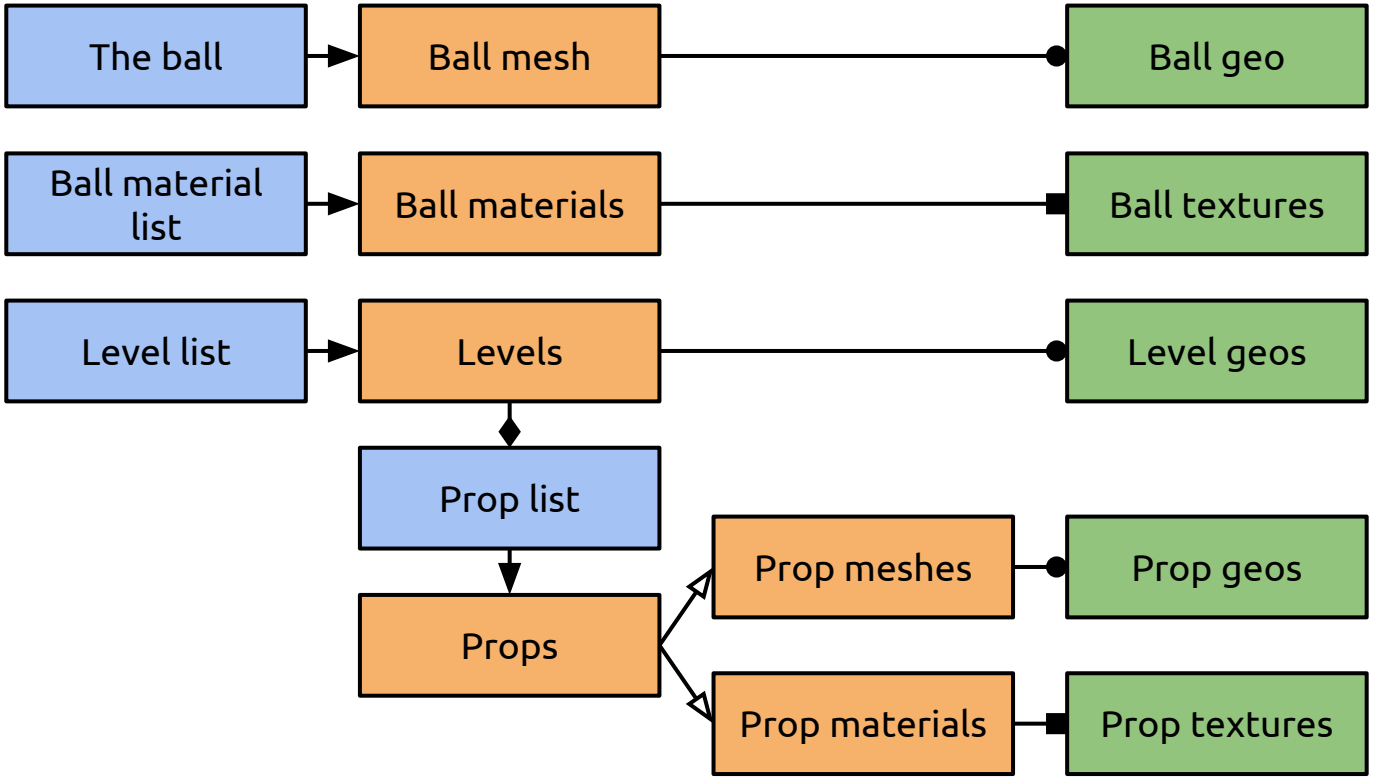
After a change

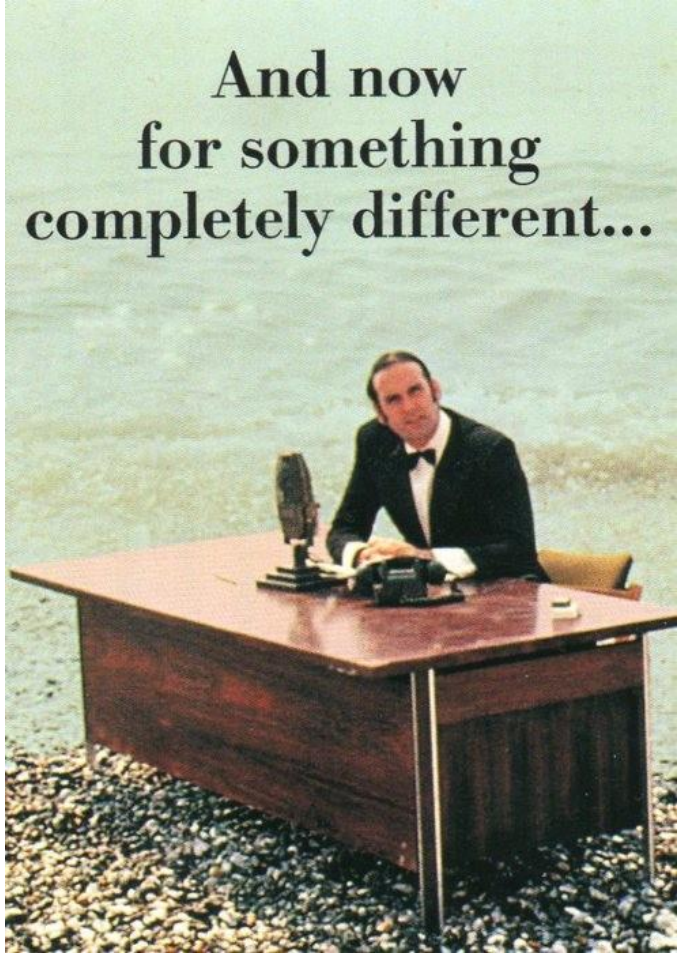


After a change



Example


The ball
Ball mesh
Ball geo
Ball material list
Ball materials
Ball textures
Level list
Levels
Level geos
Prop list
Prop meshes
Prop geos
Props
Prop materials
Prop textures


And now
for something
completely different...

Welcome to:
Build systems 437 - Experience

a.k.a. Reality
a.k.a. Learn from my many mistakes

Experience

This can be a big system

Experience

This can be a big system

- Watch your scope
- Draw a dependency diagram
- Be prepared to spend some time with it

Experience

Creating the dependency tree can take time

Experience

Creating the dependency tree can take time

- Allow the user to select a subset
- Always have your build system do the entire thing

Experience

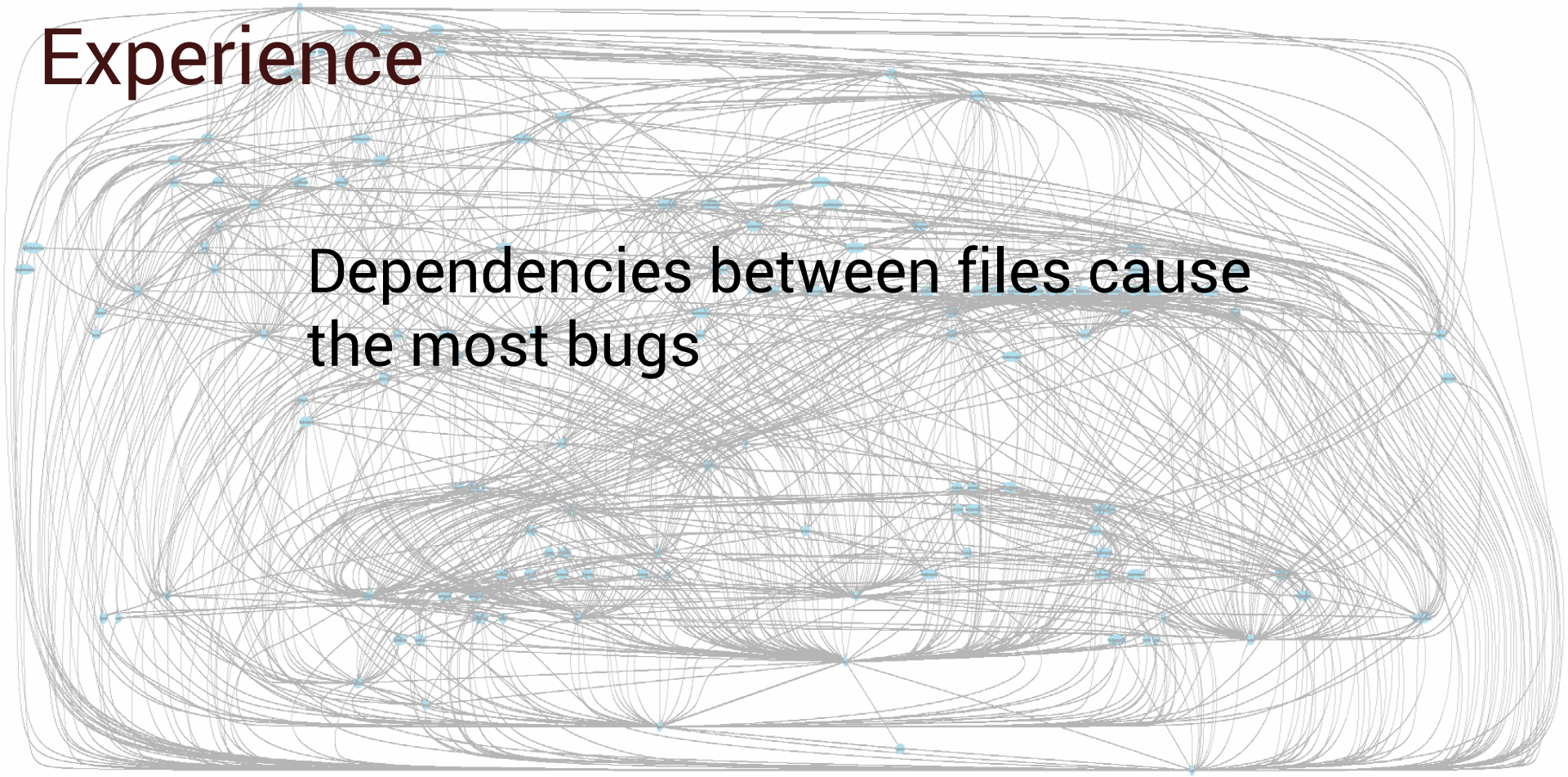
Put your intermediate files into their own directory

Experience

Put your intermediate files into their own directory

/MyProject/BuildTemp/*

Experience



Dependencies between files cause the most bugs

Experience

Dependencies between files cause the most bugs

- Missing dependencies - things don't rebuild when they should
- Extra dependencies - things build when they shouldn't

Experience

Strike a balance between export time scripts and the build

Experience

Strike a balance between export time scripts and the build

- Earlier is better for feedback
- Later is less expensive

Experience

Be explicit with what you include in the build

Experience

Be explicit with what you include in the build

- Avoid directory scans

</how does a build system work>

Sources-Processing-Targets

Dependency tracking

Change detection

- intermission -

Who should run this thing?

- intermission -

Who should run this thing?

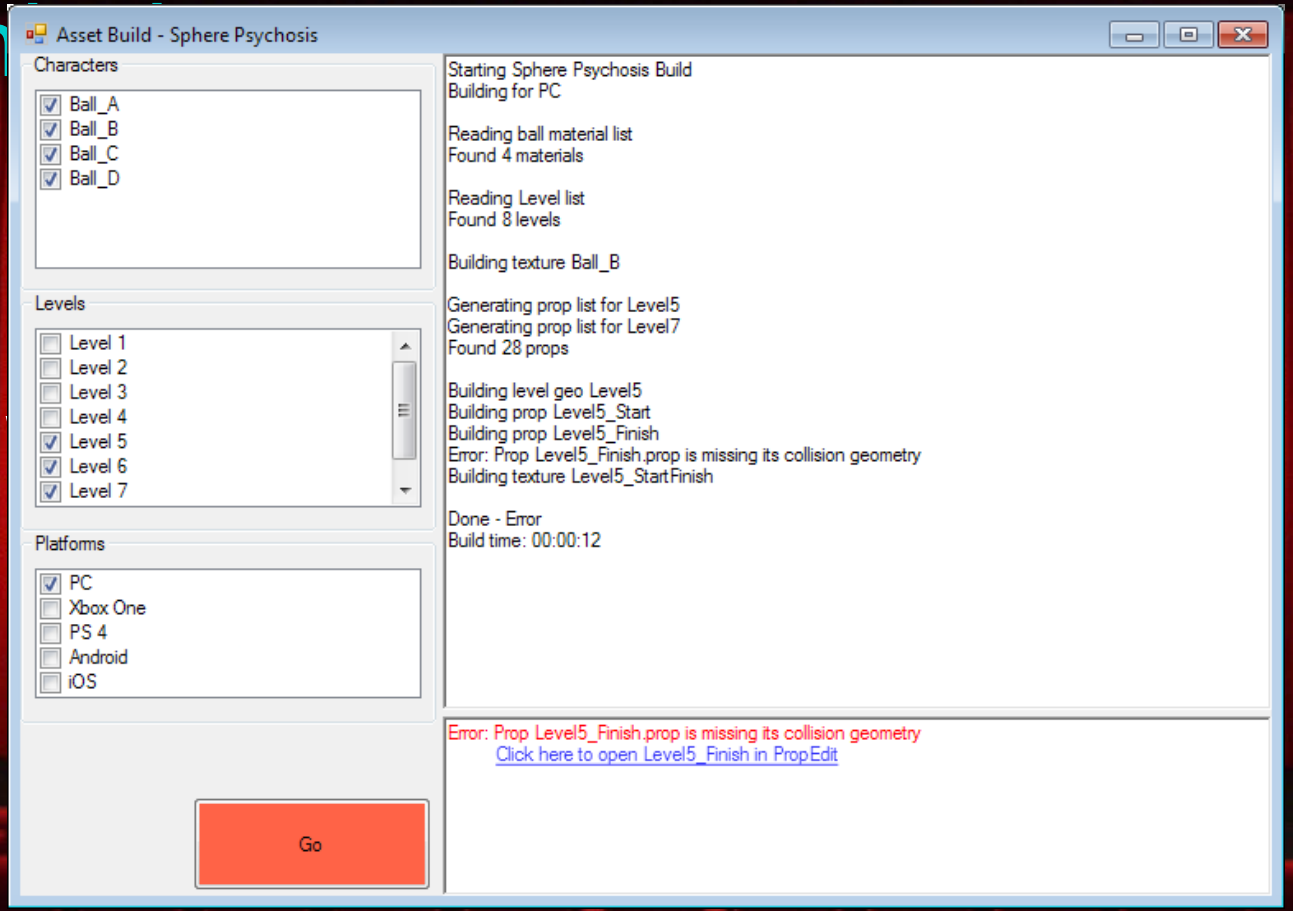
Everybody, all the time

- intermission -

Who should run this thing?

Isn’t this too complicated for artists to use?

- interm


Asset Build - Sphere Psychosis
Characters
Ball_A
Ball_B
Ball_C
Ball_D
Levels
Level 1
Level 2
Level 3
Level 4
Level 5
Level 6
Level 7
Platforms
PC
Xbox One
PS 4
Android
iOS
Go
Starting Sphere Psychosis Build
Building for PC
Reading ball material list
Found 4 materials
Reading Level list
Found 8 levels
Building texture Ball_B
Generating prop list for Level5
Generating prop list for Level7
Found 28 props
Building level geo Level5
Building prop Level5_Start
Building prop Level5_Finish
Error: Prop Level5_Finish.prop is missing its collision geometry
Building texture Level5_StartFinish
Done - Error
Build time: 00:00:12
Error: Prop Level5_Finish.prop is missing its collision geometry
Click here to open Level5_Finish in PropEdit

- intermission -

Who should run this thing?

Your continuous build system should **ALWAYS** build everything.

How do I make one?

How do I make one?

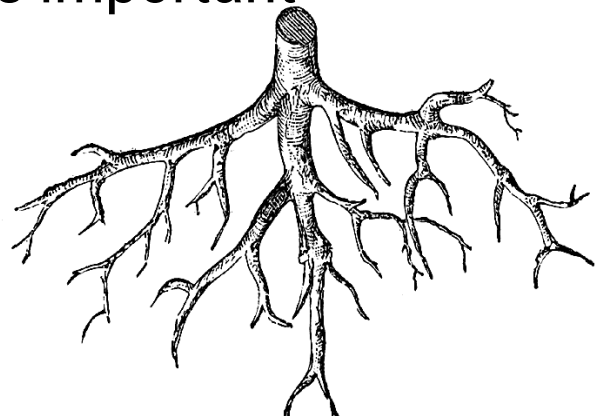
SCons

ĕs-kŏnz, skŏnz, skōnz
scons.org

SCons

What do I need to know?

The project root is important



SCons

What do I need to know?

Action = processing

Builder = SPT encapsulation

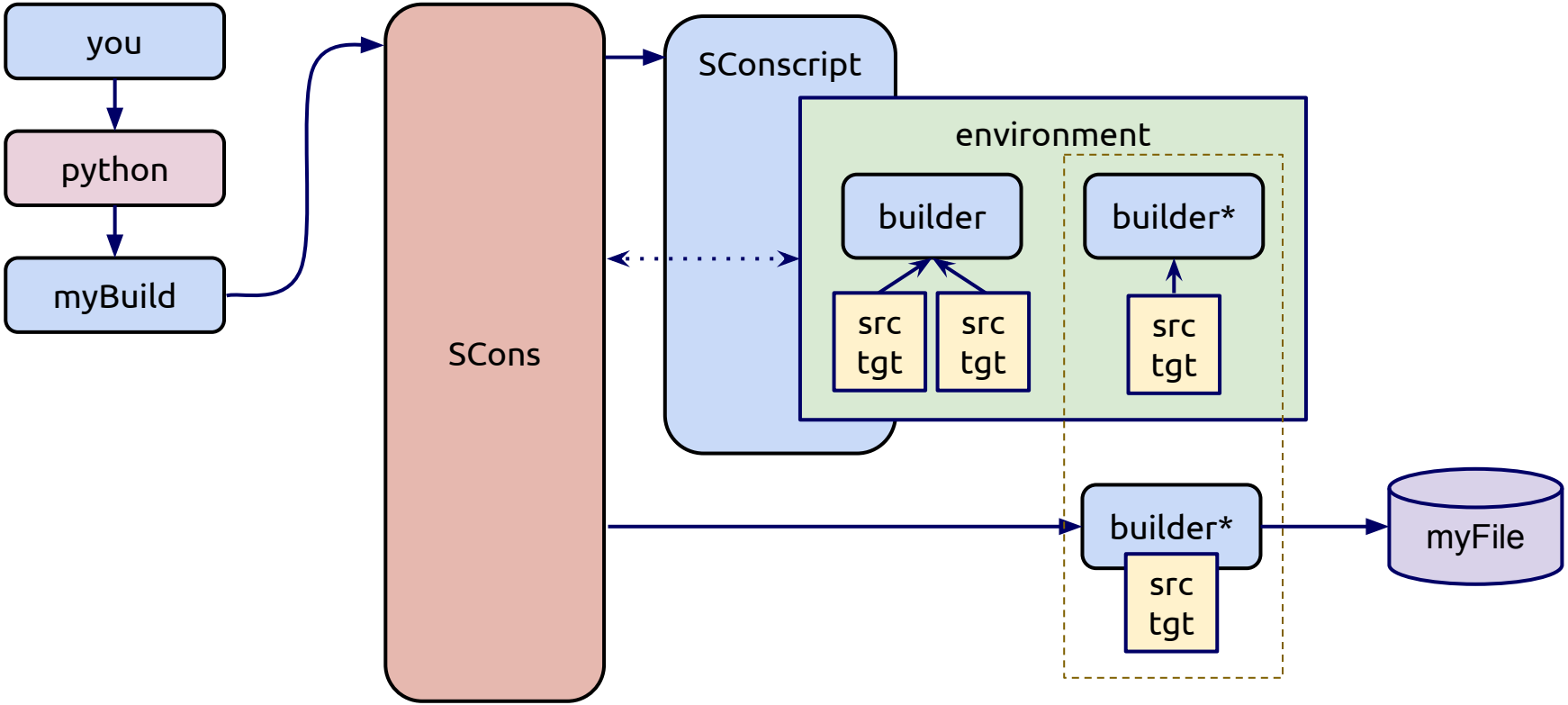
Environment = **the build**

SCons

What do I need to know?

- SCons calls you back
- Dependency setup and processing are separate

SCons


you
python
myBuild
SCons
SConscript
environment
builder
builder*
src
tgt
src
tgt
src
tgt
builder*
src
tgt
myFile

Example

<code>

Example

Replace a batch file

```
for %%F in (*.file)
do
(
    convert.exe %%~dpnxF %%~dpnF.output
)
```

Example

With SCons

Example

Create the build environment

```
env = Environment()
```

Example

Create a builder

```
env = Environment()

def my_action(target, source, env):
    convert(source[0].path, target[0].path)

env['BUILDERS']['my_build'] =
    Builder(action = my_action)
```

Example

Tell SCons about the files

```
env = Environment()

def my_action(target, source, env):
    convert(source[0].path, target[0].path)

env['BUILDERS']['my_build'] =
    Builder(action = my_action)

for src_fn in get_source_files():
    env.my_build( source = src_fn,
        target = get_target_fn(src_fn))
```

Example

Tell SCons about the files

```
env = Environment()

def my_action(target, source, env):
    convert(source[0].path, target[0].path)

env['BUILDERS']['my_build'] =
    Builder(action = my_action)

for src_fn in get_source_files():
    env.my_build( source = src_fn,
        target = get_target_fn(src_fn))
```

Done

Example

Run it

```
scons: Reading SConscript files ...
scons: done reading SConscript files.
scons: Building targets …
my_build(["myfile1.target"], ["myfile1.
source"])
.
<spew removed for sanity>
.
my_build(["myfile50000.target"],
     ["myfile50000.source"])
scons: done building targets.
Exit code:  0
```

Example

Run it again

```
scons: Reading SConscript files ...
scons: done reading SConscript files.
scons: Building targets …
scons: done building targets.
Exit code:  0
```

Dependent files

psd → tga → texture

```
for psd_fn in get_psd_files():
    tga_fn = as_tga(psd_fn)
    tex_fn = as_tex(tga_fn)
```

Dependent files

psd → tga → texture

```
for psd_fn in get_psd_files():
    tga_fn = as_tga(psd_fn)
    tex_fn = as_tex(tga_fn)

    env.tga_to_texture(tex_fn, tga_fn)
    env.psd_to_tga(tga_fn, tex_fn)
```

Running an exe

Use a generator builder

```
def my_generator(target, source, env):
    return “bin/convert.exe {0} {1}”.format(
        source[0].path, target[0].path)

env[‘BUILDERS’][‘my_build’] =
    Builder(generator = my_generator)
```

Sphere Psychosis SConscript

```
import os
import sys

import Build

# ensure_dir_exists
def ensure_dir_exists(dir):
    ext = os.path.splitext(dir)[0]
    try:
        os.makedirs(os.path.split(dir)[0] if ext else dir)
    except Exception as ex:
        pass # dir exists

# fake useful function
def fake_useful_function(target, source):
    # since we're not actually reading make sure it exists
    for src in source:
        if not os.path.exists(src.path):
            raise

    # write source list into target
    for tgt in target:
        ensure_dir_exists(tgt.path)
        with open(tgt.path, "w") as f:
            for s in source:
                f.write(s.path)


# build_mesh
def build_mesh(target, source, env) :
    fake_useful_function(target, source)

def get_mesh_target(source):
    return build_config.in_output(source, ext="geo")


# build_material
def build_material(target, source, env) :
    fake_useful_function(target, source)

def get_material_target( source ):
    return build_config.in_output(source, ext="tex")


# build_prop_list
def build_level_prop_list(target, source, env) :
    # pretend like this is opening on the complicated level file and just writing out the props
    with open(source[0].path, "r") as f:
        ensure_dir_exists(target[0].path)
        with open(target[0].path, "w") as outf:
            outf.write(f.read())

def get_proplist_target(source):
    return build_config.in_intermediate(source, ext="proplist")


# buildload_level_prop_list
def buildload_level_prop_lists(target, source, env) :
    props = set()

    for src in [src.path for src in source]:
        with open(src, "r") as f:
            levelProps = [ l.strip() for l in f.readlines()]
            props.update( levelProps )

    dependencies = { 'mtrl': set(), 'mesh': set() }
    for prop in props:
        with open(prop, "r") as f:
            for type, name in [l.split() for l in f.readlines()]:
                dependencies[type].add(name)

    for mtrl in dependencies['mtrl']:
        env.build_material( target = get_material_target(mtrl), source = mtrl )

    for mesh in dependencies['mesh']:
        env.build_mesh( target = get_mesh_target(mesh), source = mesh )

# build_prop
def build_prop(target, source, env) :
    fake_useful_function(target, source)

# create_builders
def create_builders(env):
    def make_builder(func):
        env['BUILDERS'][func.__name__] = Builder(action = func)

    make_builder(build_mesh)
    make_builder(build_material)
    make_builder(build_level_prop_list)
    make_builder(buildload_level_prop_lists)
    make_builder(build_prop)

# set_env_defaults
def set_env_defaults(env):
    pass

# start_build
# N.B. This not executing in the same module setup_scons is
def start_build():
    """Entrypoint for our main build processing"""

    # create build environment
    env = Environment(tools=[]) # no defaults tools, loads faster
    set_env_defaults(env)

    # create builders
    create_builders(env

    # setup ball mesh build
    env.build_mesh(target = get_mesh_target(build_config.ball_mesh_fn), source = build_config.ball_mesh_fn )

    # setup ball material builds
    materials = []
    with open(build_config.ball_mtrl_list_fn) as f:
        materials = [s.strip() for s in f.readlines()]

    for material in materials:
        src_mtrl = os.path.join("art/ball", material)
        env.build_material(target = get_material_target(src_mtrl), source=src_mtrl)


    # setup level builds
    levels = []
    with open(build_config.level_list_fn) as f:
        levels = [s.strip() for s in f.readlines()]

    proplists = []
    for level in levels:
        src_level = os.path.join("art/Levels", level)
        env.build_mesh(target = get_mesh_target(src_level), source = src_level)
        proplists.append(get_proplist_target(src_level))
        env.build_level_prop_list(target = proplists[-1], source = src_level)


    # force loading and processing of prop lists
    node = env.buildload_level_prop_lists( target = "buildload_level_prop_lists.dummytarget", source = proplis
    env.AlwaysBuild( node )

# entrypoint
build_config = sys.modules['__main__'].build_config
start_build()
```

SCons - other fun stuff

SCons - other fun stuff

Multicore support

```
num_cpu = int(os.environ.get('NUM_CPU', 2))
SetOption('num_jobs', num_cpu)
```

SCons - other fun stuff

Build cache

Why build it if someone else already has?

```
CacheDir('//server/MyProject/BuildCache')

# opting out
NoCache('myfile.file')
```

SCons - other fun stuff

Incredibuild

```
BuildConsole /command="python MyBuild.py"
```

SCons - other fun stuff

*The rest of SCons

SCons - gotcha’s

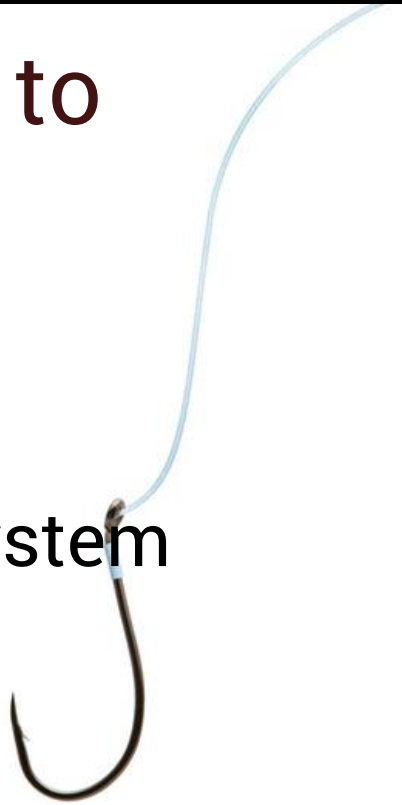
*check notes

</how do I make one>

Other interesting things you can to with a robust build system

Other interesting things you can to with a robust build system

Hook it into your asset reload system



Other interesting things you can to with a robust build system

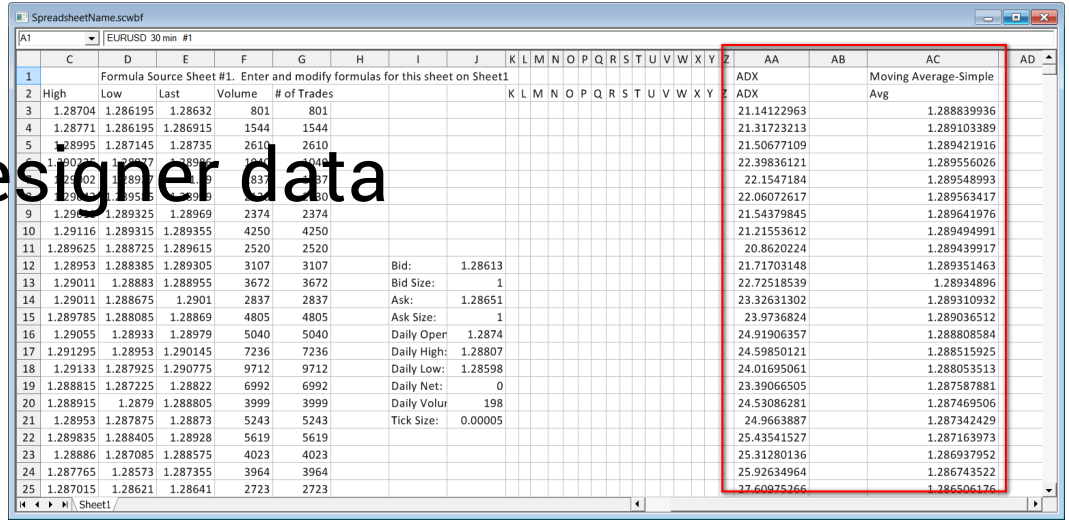
Asset validation

Other interesting things you can to with a robust build system

Asset DB generation

Other interesting things you can to with a robust build system

Validate designer data



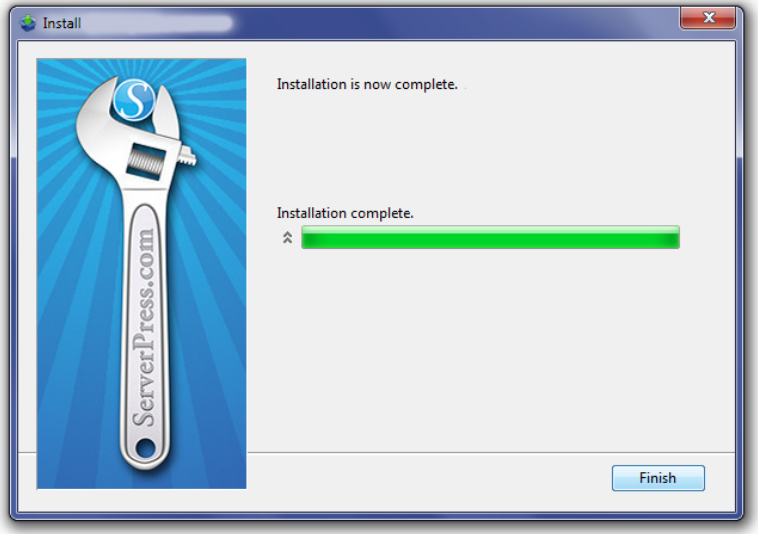
	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD
1																									ADX		Moving Average-Simple	
2	High	Low	Last	Volume	# of Trades																				ADX		Avg	
3	1.28704	1.286195	1.28632	801	801																				21.14122963		1.288839936	
4	1.28771	1.286195	1.286915	1544	1544																				21.31723213		1.289103389	
5	1.28995	1.287145	1.28735	2610	2610																				21.50677109		1.289421916	
6	1.290035	1.28877	1.289865	1040	1040																				22.39836121		1.289556026	
7	1.29002	1.28917	[illegible]	837	837																				22.1547184		1.289548993	
8	1.29[illegible]	1.29513	[illegible]	[illegible]	[illegible]																				22.06072617		1.289563417	
9	1.29[illegible]	1.289325	1.28969	2374	2374																				21.54379845		1.289641976	
10	1.29116	1.289315	1.289355	4250	4250																				21.21553612		1.289494991	
11	1.289625	1.288725	1.289615	2520	2520																				20.8620224		1.289439917	
12	1.28953	1.288385	1.289305	3107	3107		Bid:	1.28613																	21.71703148		1.289351463	
13	1.29011	1.28883	1.288955	3672	3672		Bid Size:	1																	22.72518539		1.28934896	
14	1.29011	1.288675	1.2901	2837	2837		Ask:	1.28651																	23.32631302		1.289310932	
15	1.289785	1.288085	1.28869	4805	4805		Ask Size:	1																	23.9736824		1.289036512	
16	1.29055	1.28933	1.28979	5040	5040		Daily Oper	1.2874																	24.91906357		1.288808584	
17	1.291295	1.28953	1.290145	7236	7236		Daily High:	1.28807																	24.59850121		1.288515925	
18	1.29133	1.287925	1.290775	9712	9712		Daily Low:	1.28598																	24.01695061		1.288053513	
19	1.288815	1.287225	1.28822	6992	6992		Daily Net:	0																	23.39066505		1.287587881	
20	1.288915	1.2879	1.288805	3999	3999		Daily Volur	198																	24.53086281		1.287469506	
21	1.28953	1.287875	1.28873	5243	5243		Tick Size:	0.00005																	24.9663887		1.287342429	
22	1.289835	1.288405	1.28928	5619	5619																				25.43541527		1.287163973	
23	1.28886	1.287085	1.288575	4023	4023																				25.31280136		1.286937952	
24	1.287765	1.28573	1.287355	3964	3964																				25.92634964		1.286743522	
25	1.287015	1.28621	1.28641	2723	2723																				27.60975266		1.286506176	

Other interesting things you can to with a robust build system

Automated smoke and unit tests

Other interesting things you can to with a robust build system

Tool install

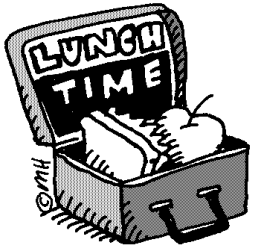


Other interesting things you can to with a robust build system

Code build

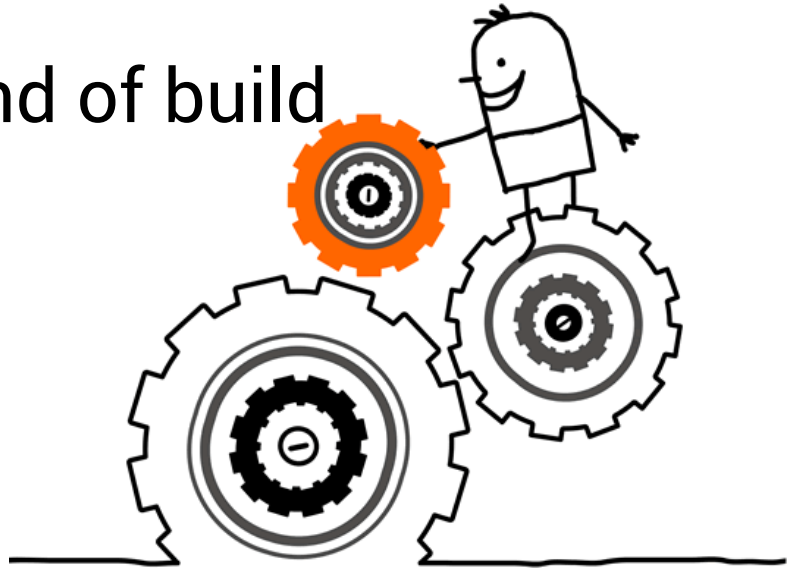
Final thoughts

...and then lunch



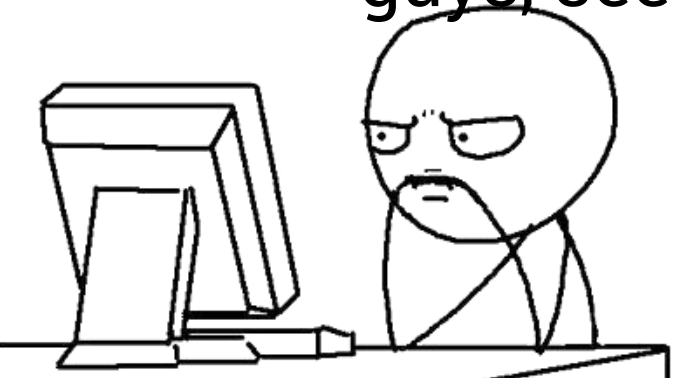
Final thoughts

It’s 2014, use some kind of build system



Final thoughts

Talk to your coders / build / deploy guys, see what they've got going on

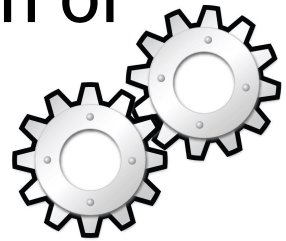


Final thoughts

If you're comfortable with Python, have a look at SCons

Final thoughts

Start small, with a single system or process...



Final thoughts

Think big, you can do a huge amount of stuff with a build system

Final thoughts

Do not allow assets to break the build, leave that to the coders

Thanks

Next Level Games
Sony Online Entertainment

Jim Randall

Asset Build Systems

Kris Lang
Client Technical Director
Game Technology Group @ SOE

GAME DEVELOPERS CONFERENCE®
SAN FRANCISCO, CA
MARCH 17–21, 2014
EXPO DATES: MARCH 19–21
2014