accu 2022

ADVANCED DEPENDENCIES MODEL

IN CONAN 2.0 C, C++ PACKAGE MANAGER

DIEGO RODRIGUEZ-LOSADA

Who am I

JFrog | CONAN C/C++ package manager

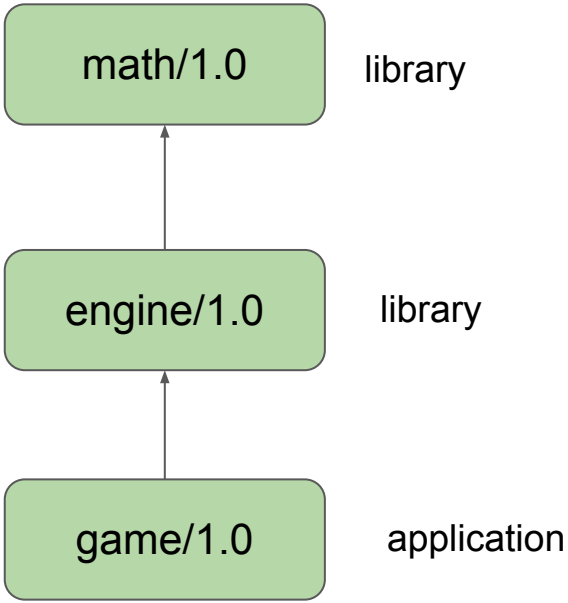
What is Conan

- FOSS Package Manager for C/C++
  - MIT license
- Universal
  - Multi-platform
  - Any build system
- Efficient - Binaries
- Stable, active and supported

Outline

- **The C and C++ native artifacts model**
- Conan 1.X packages and dependencies model
- Limitations of Conan 1.X model
- Conan 2.0 new model
- Demo
- Conclusion

C/C++ native artifacts model



Math & Engine: Static libraries (.lib, .a)

math2.cpp

```
int add(int a, int b){
  return a + b;
}
```

engine.cpp

```
#include "math2.h"
int move3d(int x, int y, int z){
    return add(x, add(y, z));
}
```

math2.lib

```
?add@@YAHHH@Z (int __cdecl add(int,int)):
  ...00000: 89 54 24 10      mov     dword ptr [rsp+10h],edx
  ...00004: 89 4C 24 08      mov     dword ptr [rsp+8],ecx
  ...00008: 57               push    rdi
  ...00009: 8B 44 24 18      mov     eax,dword ptr [rsp+18h]
  ...0000D: 8B 4C 24 10      mov     ecx,dword ptr [rsp+10h]
  ...00011: 03 C8            add     ecx,eax
  ...
```

engine.lib

```
?move3d@@YAHHHH@Z (int __cdecl move3d(int,int,int)):
  ...
  ...00021: 8B 4C 24 30      mov     ecx,dword ptr [rsp+30h]
  ...00025: 8B 54 24 40      mov     edx,dword ptr [rsp+40h]
  ...00029: 8B 4C 24 38      mov     ecx,dword ptr [rsp+38h]
  ...0002D: E8 00 00 00 00   call    ?add@@YAHHH@Z
  ...00032: 8B D0            mov     edx,eax
  ...00034: 8B 4C 24 30      mov     ecx,dword ptr [rsp+30h]
  ...00038: E8 00 00 00 00   call    ?add@@YAHHH@Z
  ...
```

Executable linking static libraries

game.cpp

```
#include "engine.h"

int main(){
 std::cout << move3d(1, 2, 3);
}
```

game.exe

```
main:
  ...
  ...01400014F5: 41 B8 80 0D 00 00  mov       r8d,0D80h
  ...01400014FB: BA 29 09 00 00     mov       edx,929h
  ...0140001500: B9 D2 04 00 00     mov       ecx,4D2h
  ...0140001505: E8 8B FC FF FF     call  ?move3d@@YAHHHH@Z

?move3d@@YAHHHH@Z (int __cdecl move3d(int,int,int)):
  ...0002D: E8 00 00 00 00  call      ?add@@YAHHH@Z
  ...00032: 8B D0           mov       edx,eax
  ...00034: 8B 4C 24 30     mov       ecx,dword ptr [rsp+30h]
  ...00038: E8 00 00 00 00  call      ?add@@YAHHH@Z

?add@@YAHHH@Z (int __cdecl add(int,int)):
  ...00000: 89 54 24 10     mov       dword ptr [rsp+10h],edx
```

Shared library linking static

math2.cpp

```
int add(int a, int b){
  return a + b;
}
```

math2.lib

```
?add@@YAHHH@Z (int __cdecl add(int,int)):
  ...
  ...00011: 03 C8              add         ecx,eax
  ...
```

engine.cpp

```
#include "math2.h"
int move3d(int x, int y, int z){
    return add(x, add(y, z));
}
```

engine.dll

```
?move3d@@YAHHHH@Z (int __cdecl move3d(int,int,int)):
  ...
  ...00021: 8B 4C 24 30        mov         ecx,dword ptr [rsp+30h]
  ...00025: 8B 54 24 40        mov         edx,dword ptr [rsp+40h]
  ...00029: 8B 4C 24 38        mov         ecx,dword ptr [rsp+38h]
  ...0002D: E8 00 00 00 00     call        ?add@@YAHHH@Z
  ...00032: 8B D0              mov         edx,eax
  ...00034: 8B 4C 24 30        mov         ecx,dword ptr [rsp+30h]
  ...00038: E8 00 00 00 00     call        ?add@@YAHHH@Z
?add@@YAHHH@Z (int __cdecl add(int,int)):
  ...
  ...00011: 03 C8              add         ecx,eax
```

Executable linking shared library

game.cpp

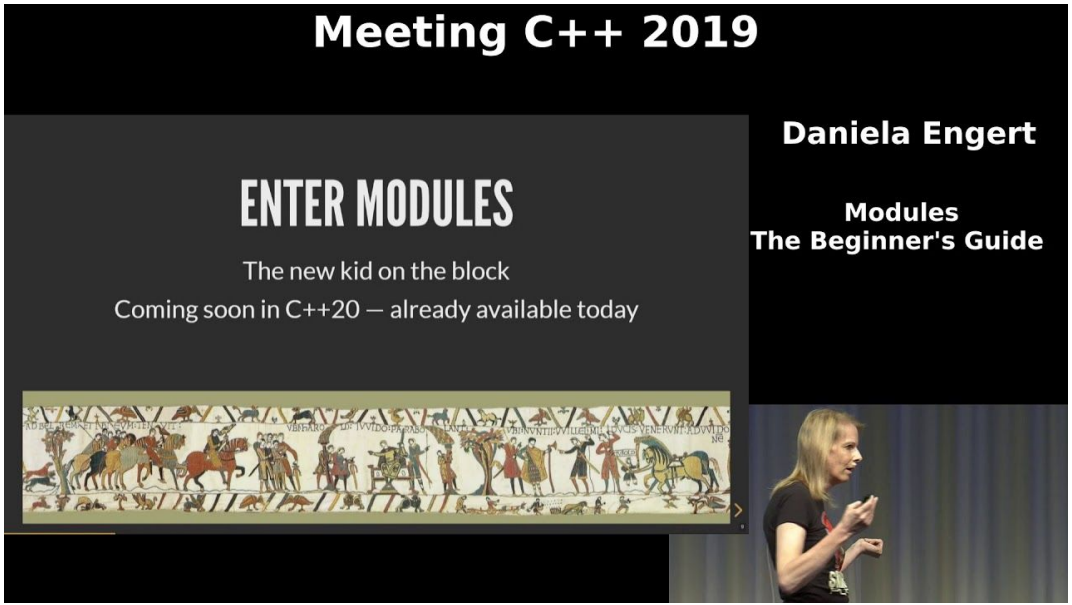
```
#include "engine.h"

int main(){
 std::cout << move3d(1, 2, 3);
}
```



```
main:
  ...
  ...01400014F5: 41 B8 80 0D 00 00  mov     r8d,0D80h
  ...01400014FB: BA 29 09 00 00     mov     edx,929h
  ...0140001500: B9 D2 04 00 00     mov     ecx,4D2h
  ...0140001505: E8 8B FC FF FF     call  ?move3d@@YAHHHH@Z

?move3d@@YAHHHH@Z:
…140001EC9: FF 25 01 D4 00 00     jmp     qword ptr
                                  [__imp_?move3d@@YAHHHH@Z]
```

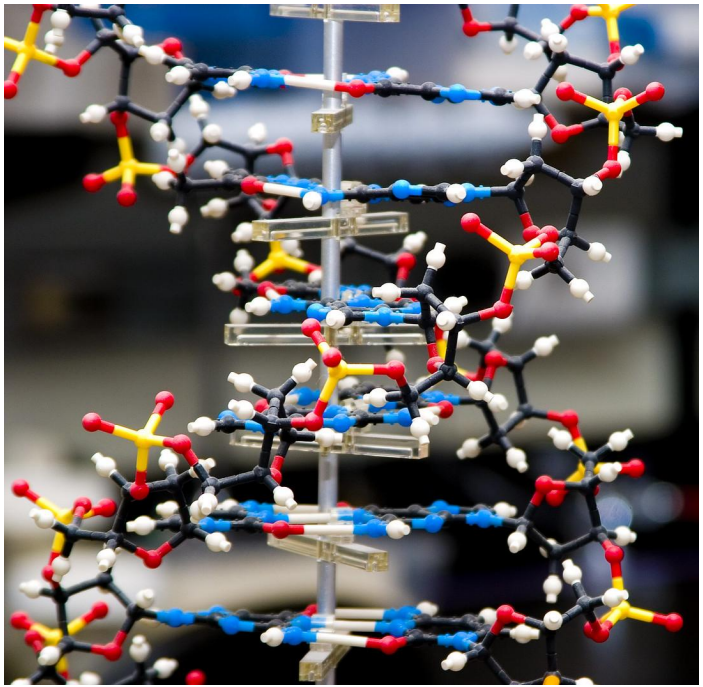
C++20 modules do not change this

- Shared, static libraries still exist

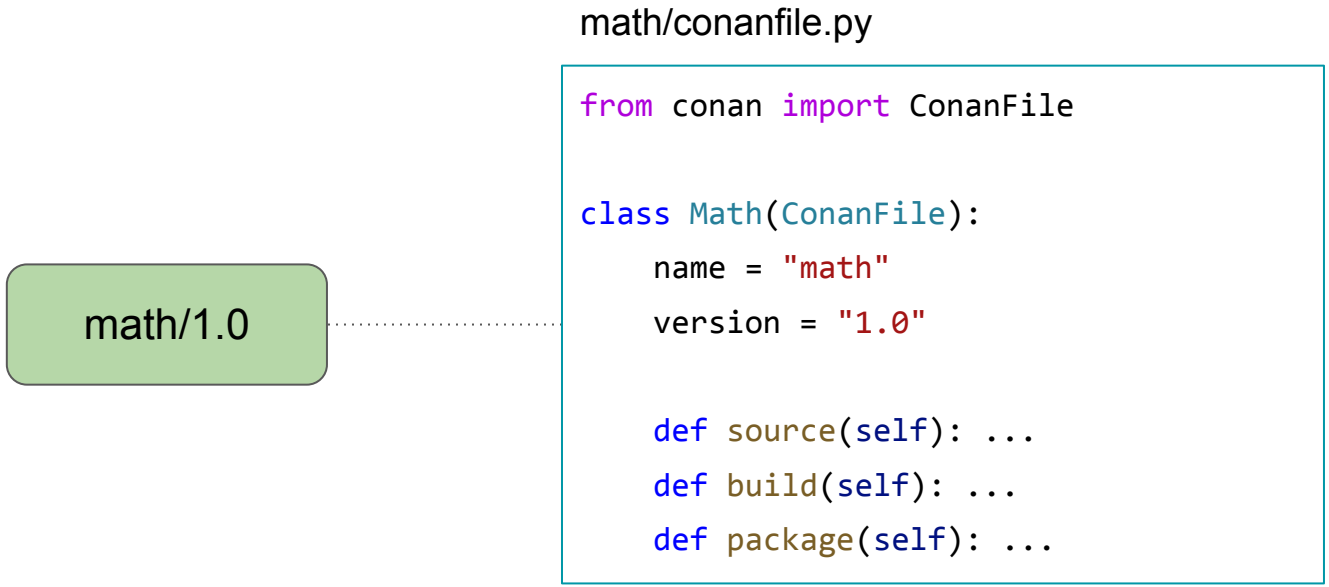


Outline

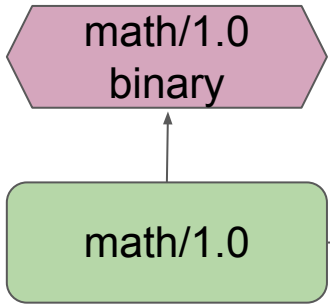
- The C and C++ native artifacts model
- **Conan 1.X packages and dependencies model**
  - **Packages and requirements**
  - Binary model
  - Tool requires and contexts
- Limitations of Conan 1.X model
- Conan 2.0 new model
- Demo
- Conclusion



Conanfile: A package “recipe”



Conanfile: A package “recipe”




```
$ git clone … math && cd math
$ conan create .
```

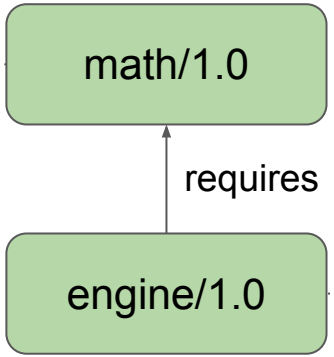
math/conanfile.py

```python
from conan import ConanFile

class Math(ConanFile):
    name = "math"
    version = "1.0"

    def source(self): ...
    def build(self): ...
    def package(self): ...
```

Conan 1.X dependency model

math/conanfile.py

```
from conan import ConanFile

class Math(ConanFile):
    name = "math"
    version = "1.0"
```



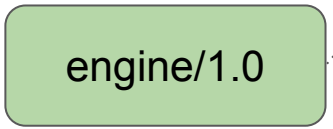
engine/conanfile.py

```
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"
    generators = "CMakeDeps"
    requires = "math/1.0"

    # Or alternatively, if logic needed
    def requirements(self):
        if <something>:
            self.requires("math/1.0")
```

Conan 1.X dependency model

engine/conanfile.py

```python
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"
    generators = "CMakeDeps"
    requires = "math/1.0"
```

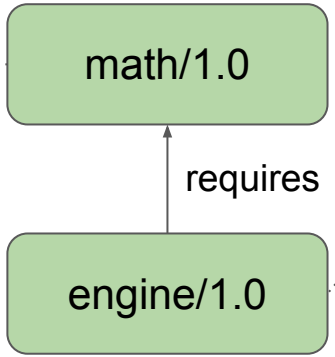



```
$ git clone … engine && cd engine
$ conan install .
```

Conan 1.X dependency model

math/conanfile.py

```
from conan import ConanFile

class Math(ConanFile):
    name = "math"
    version = "1.0"
```



engine/conanfile.py

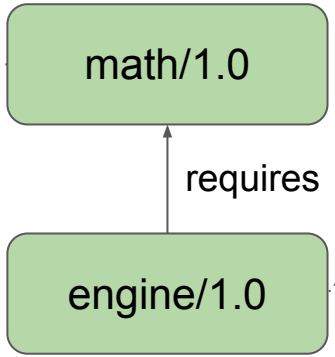
```
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"
    generators = "CMakeDeps"
    requires = "math/1.0"
```

```
$ git clone … engine && cd engine
$ conan install .
```

Conan 1.X dependency model

math/conanfile.py

```
from conan import ConanFile

class Math(ConanFile):
    name = "math"
    version = "1.0"
```



engine/conanfile.py

```
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"
    generators = "CMakeDeps"
    requires = "math/1.0"
```

```
$ git clone … engine && cd engine
$ conan install .
```

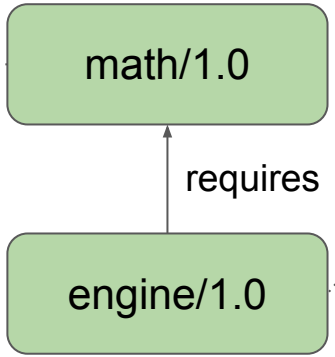
math-config.cmake

```
set_property(TARGET math::math …)
```

Conan 1.X dependency model

math/conanfile.py

```
from conan import ConanFile

class Math(ConanFile):
    name = "math"
    version = "1.0"
```



engine/conanfile.py

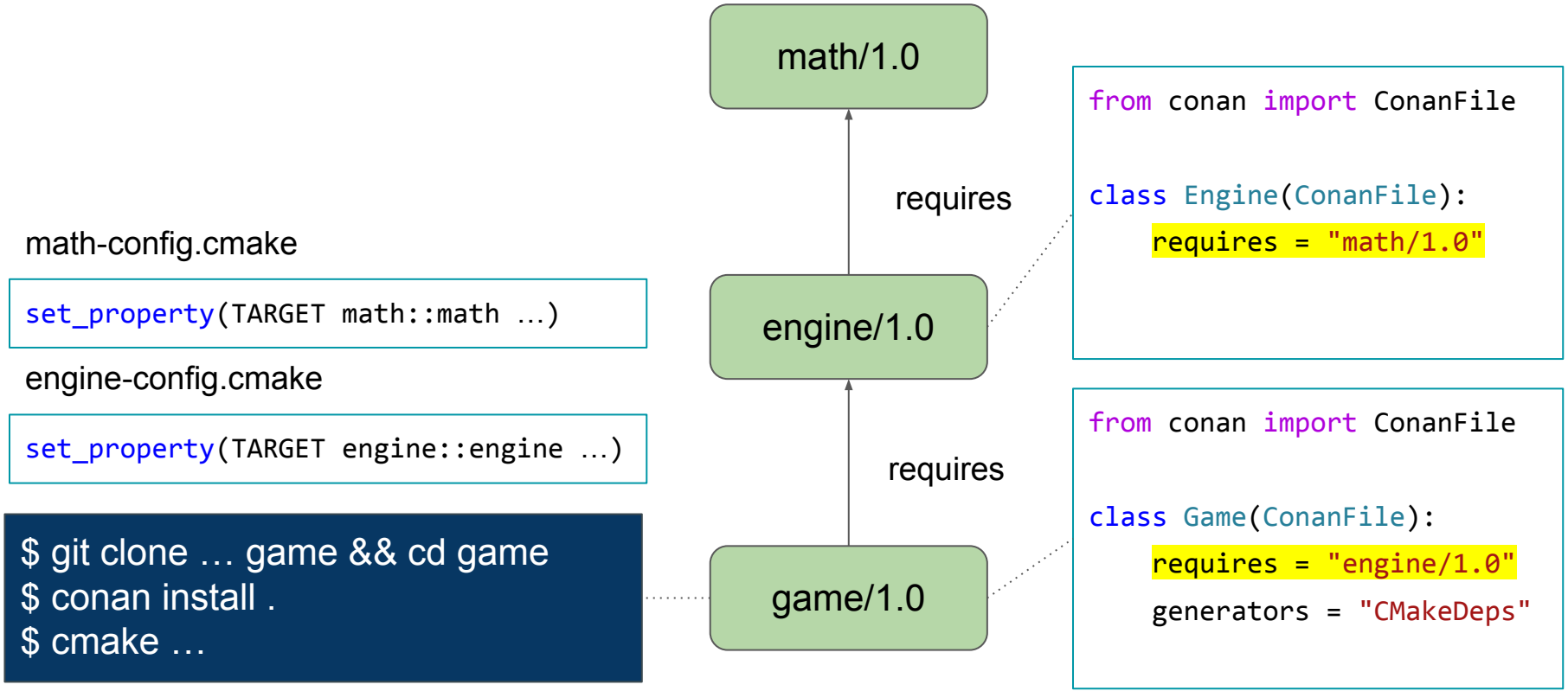
```
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"
    generators = "CMakeDeps"
    requires = "math/1.0"
```

```
$ git clone … engine && cd engine
$ conan install .
$ cmake … # engine/CMakeLists.txt
```

math-config.cmake

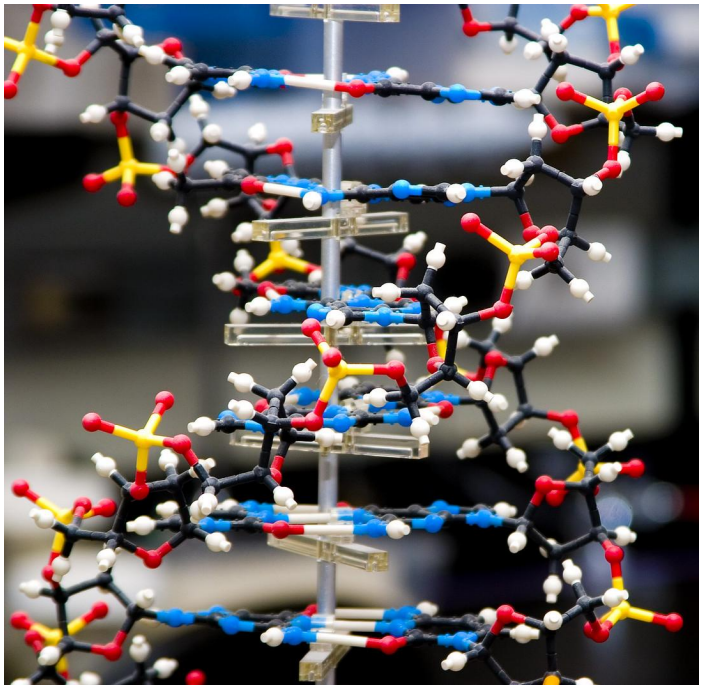
```
set_property(TARGET math::math …)
```

Conan 1.X dependency model: Transitive deps

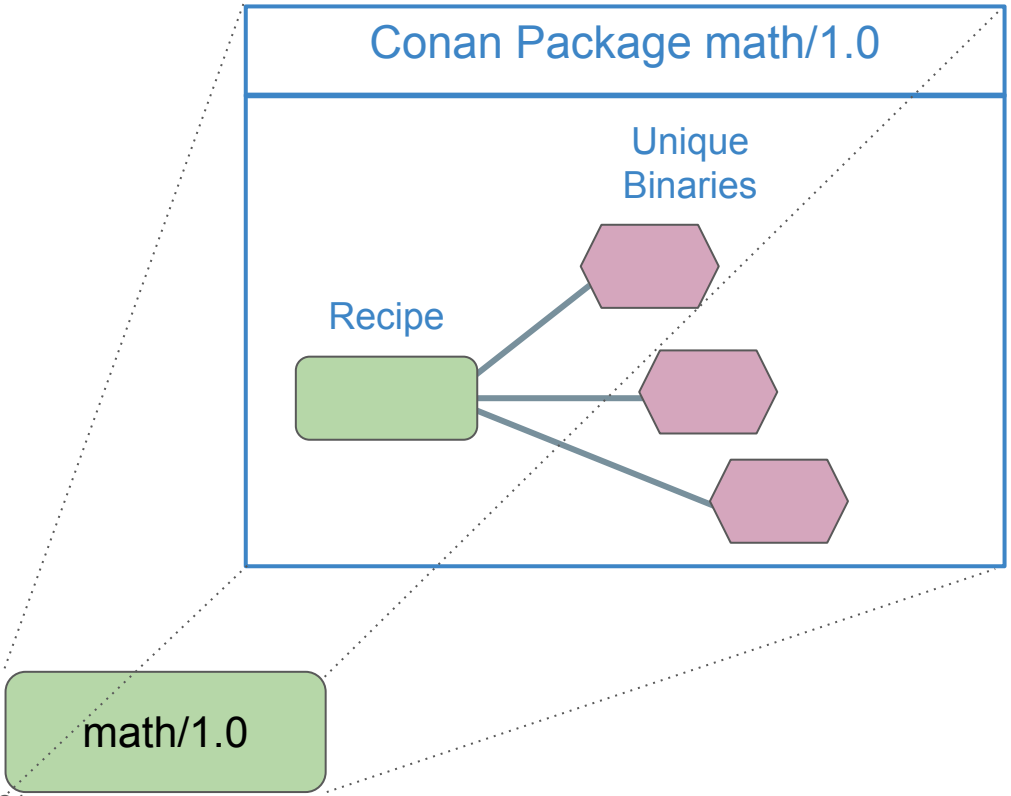


Outline

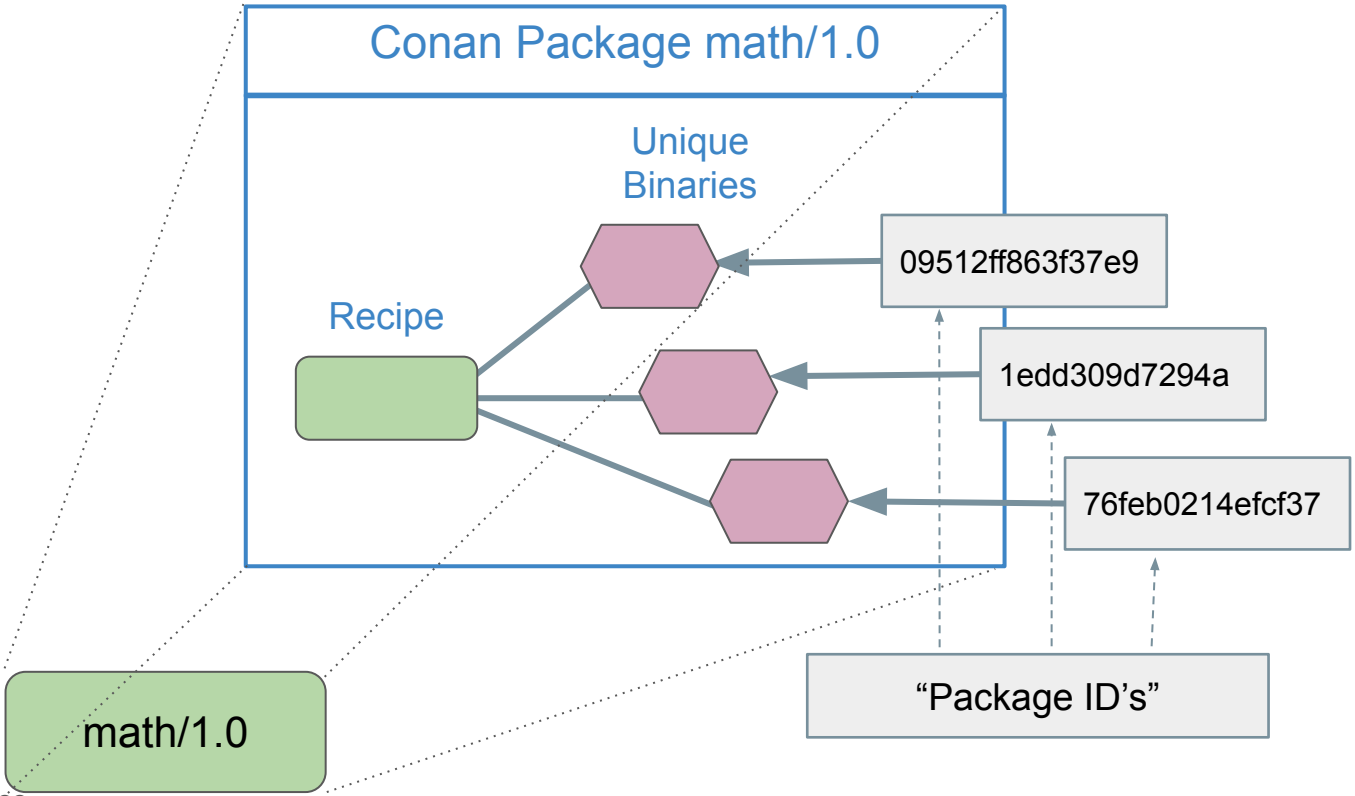
- The C and C++ native artifacts model
- **Conan 1.X packages and dependencies model**
  - Packages and requirements
  - **Binary model**
  - Tool requires and contexts
- Limitations of Conan 1.X model
- Conan 2.0 new model
- Demo
- Conclusion, Q&A



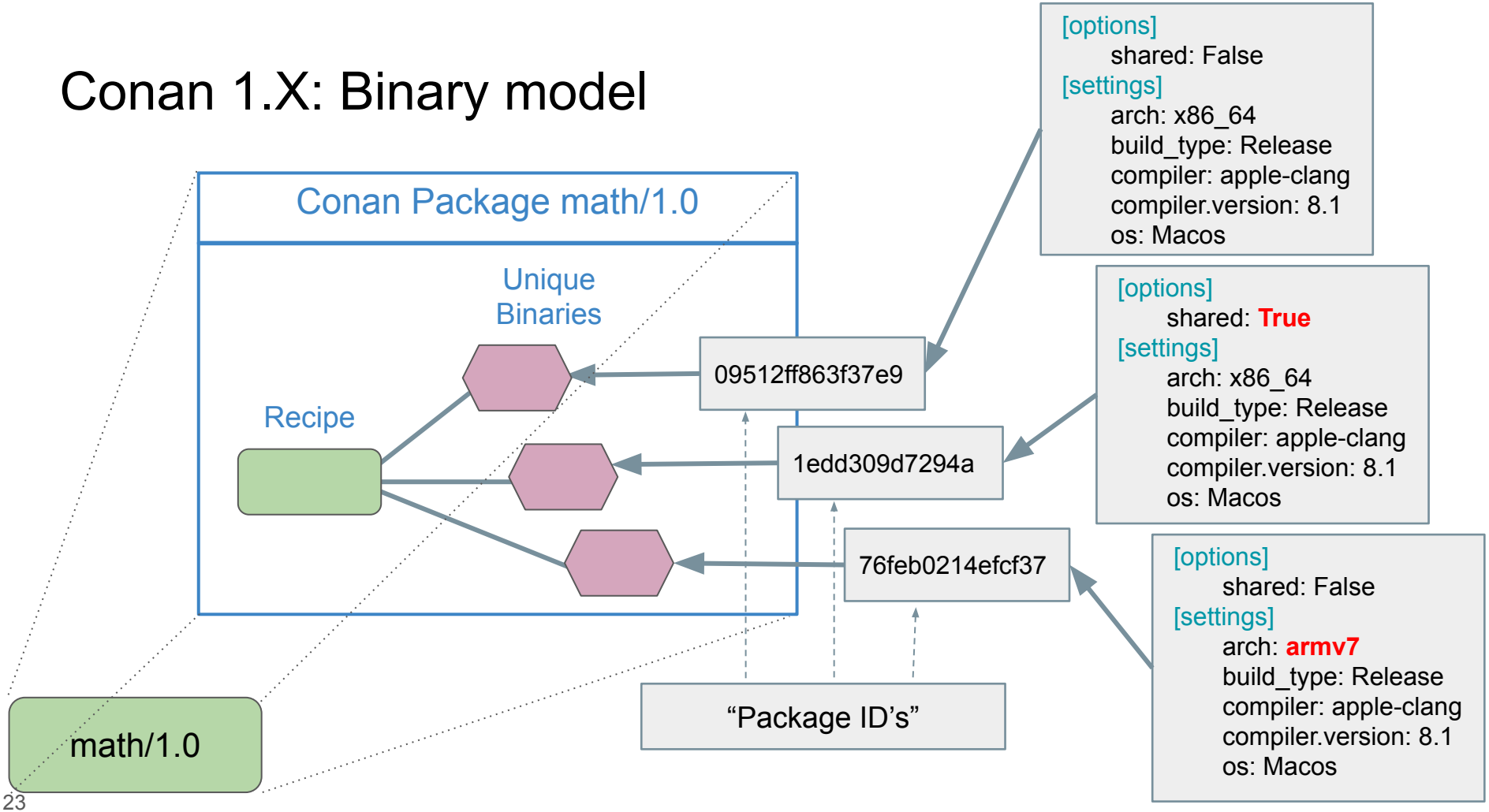
Conan 1.X: Binary model



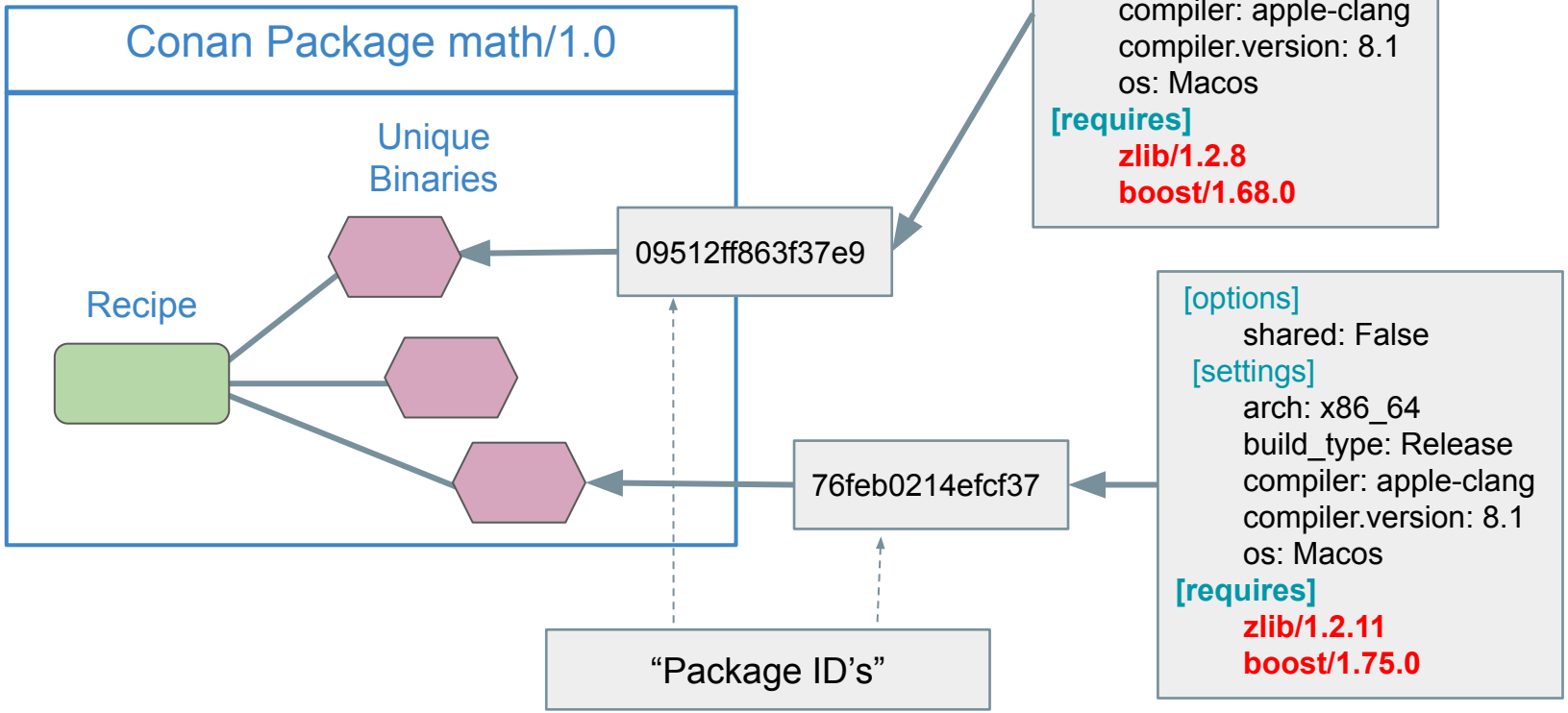
Conan 1.X: Binary model



Conan 1.X: Binary model

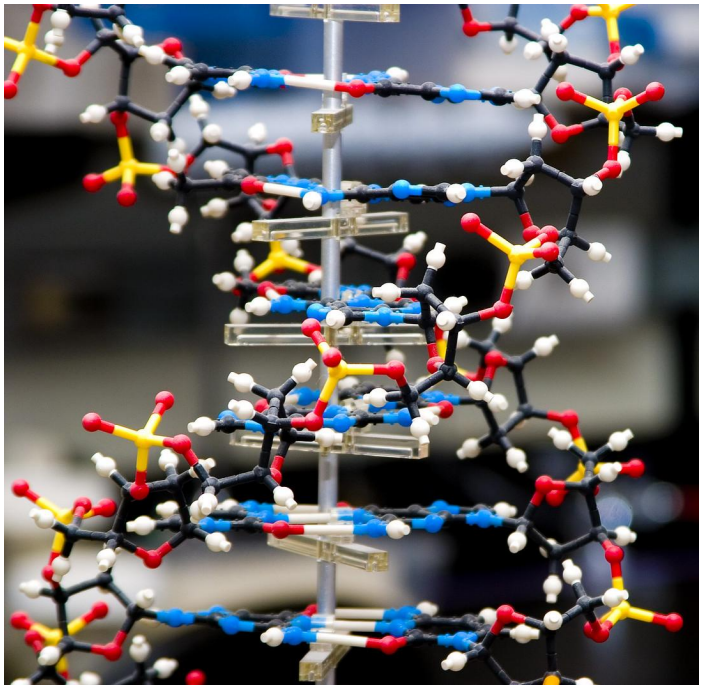


Conan 1.X: Full Binary model

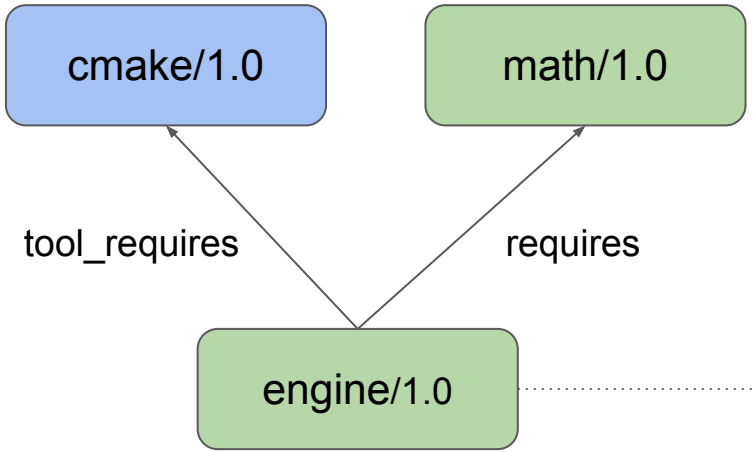


Outline

- The C and C++ native artifacts model
- **Conan 1.X packages and dependencies model**
  - Packages and requirements
  - Binary model
  - **Tool requires and contexts**
- Limitations of Conan 1.X model
- Conan 2.0 new model
- Demo
- Conclusion, Q&A

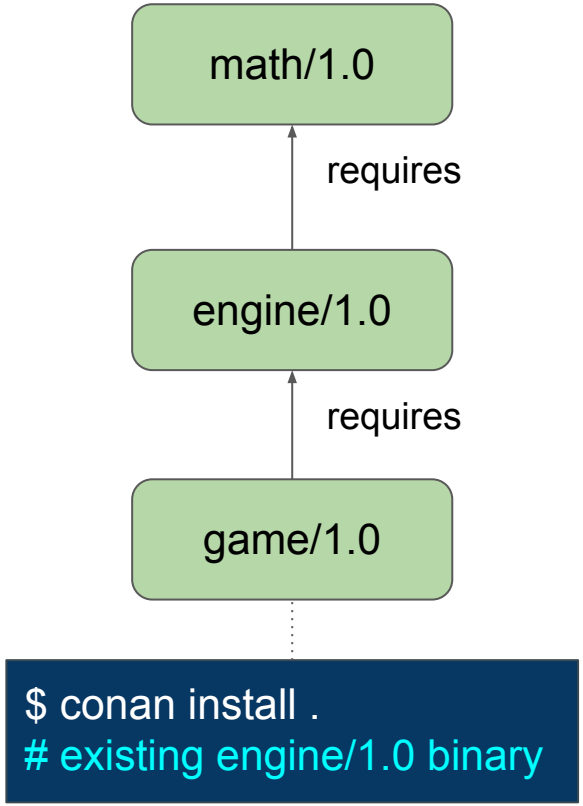


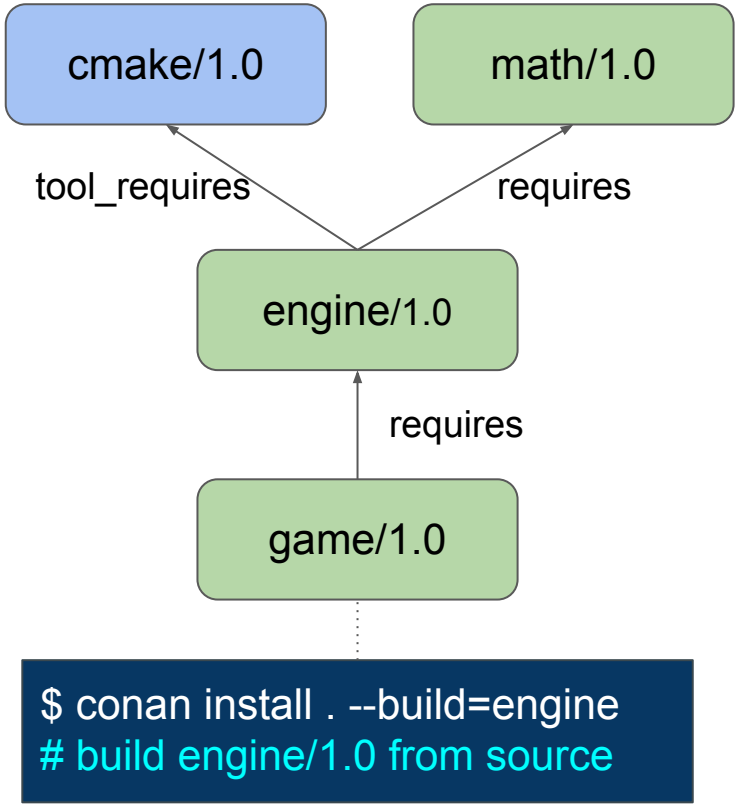
Conan 1.X: tool_requires



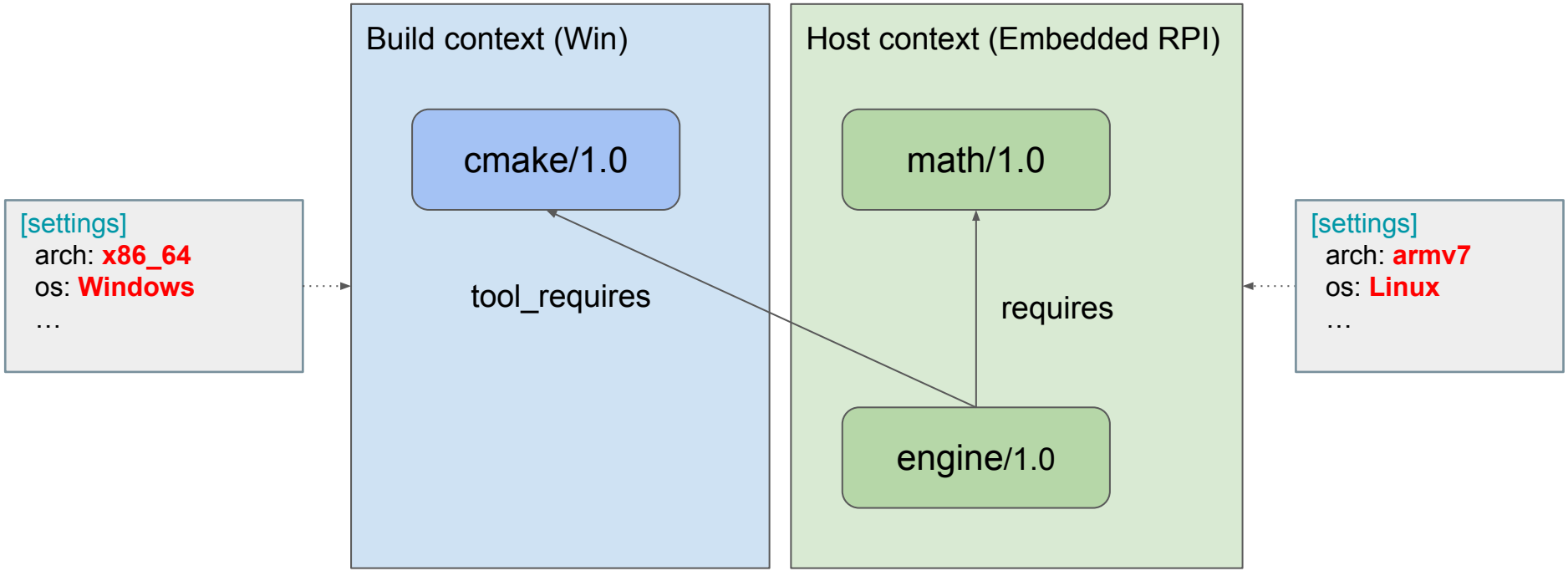

```python
from conan import ConanFile

class Engine(ConanFile):
    requires = "math/1.0"
    tool_requires = "cmake/1.0"
```

Conan 1.X: tool_requires

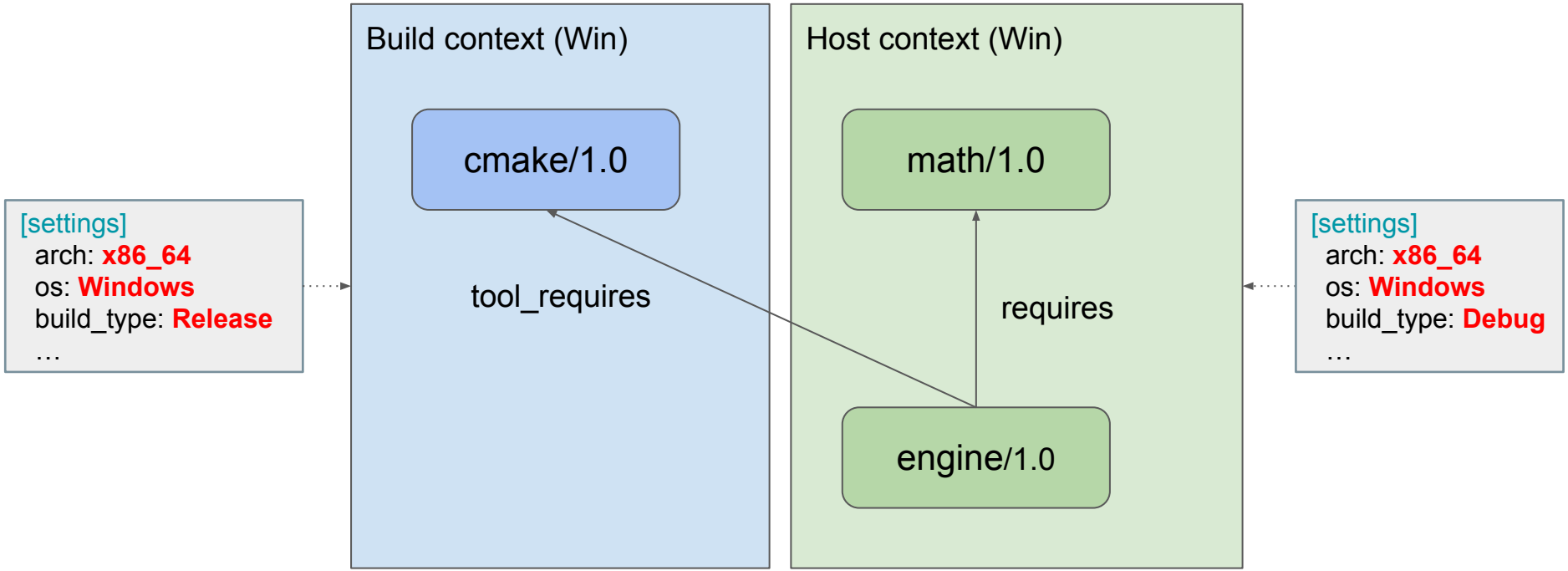




Conan 1.X: Build and host contexts: cross build model



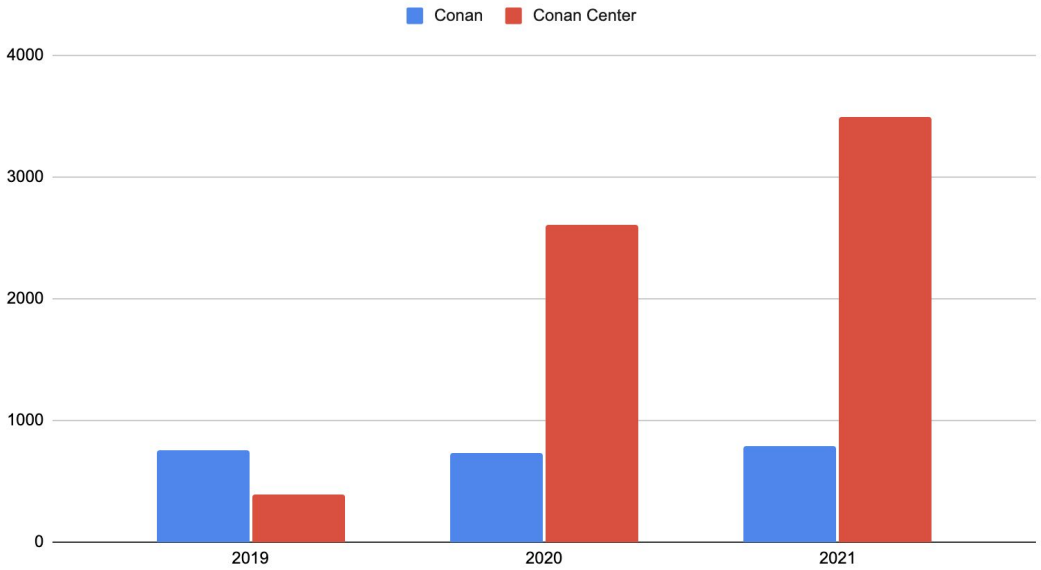
Conan 1.X: Contexts: not only for cross build



Outline

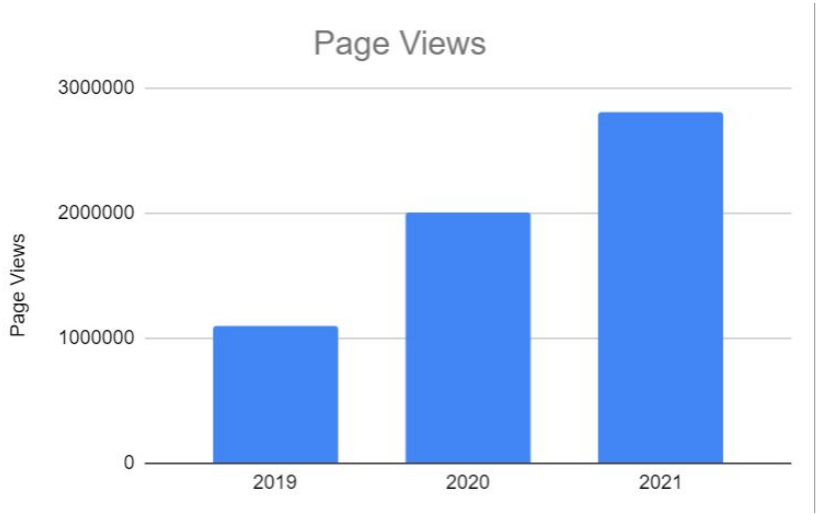
- The C and C++ native artifacts model
- Conan 1.X packages and dependencies model
  - Packages and requirements
  - Binary model
  - Tool requires and contexts
- **Limitations of Conan 1.X model**
- Conan 2.0 new model
- Demo
- Conclusion, Q&A

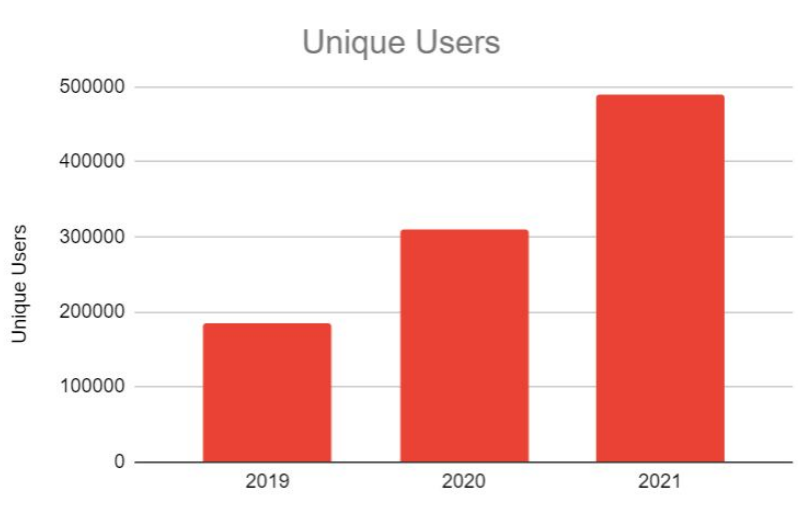
of pull requests / year



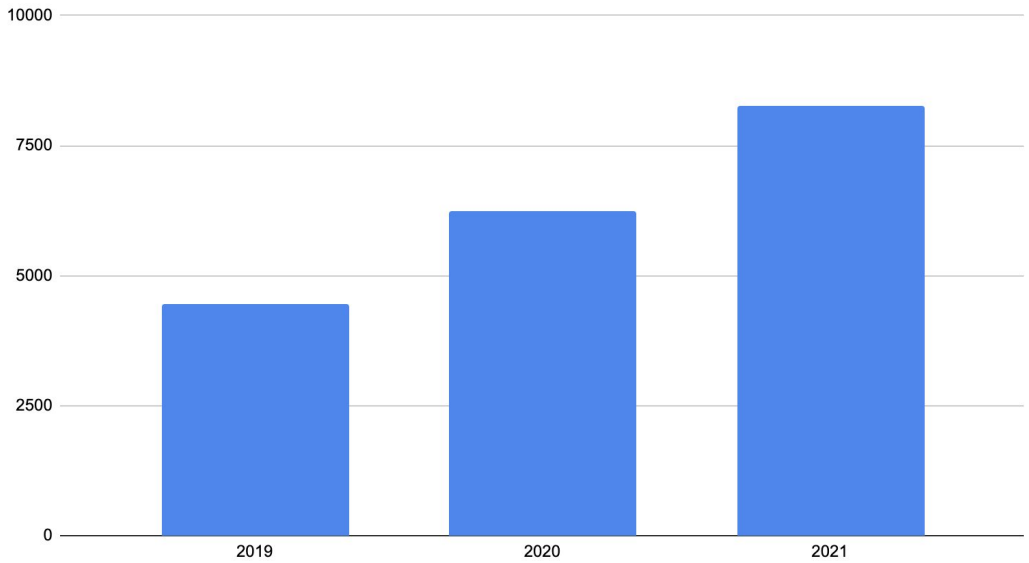
https://isocpp.org/files/papers/CppDevSurvey-2021-04-summary.pdf

Website views & users / year





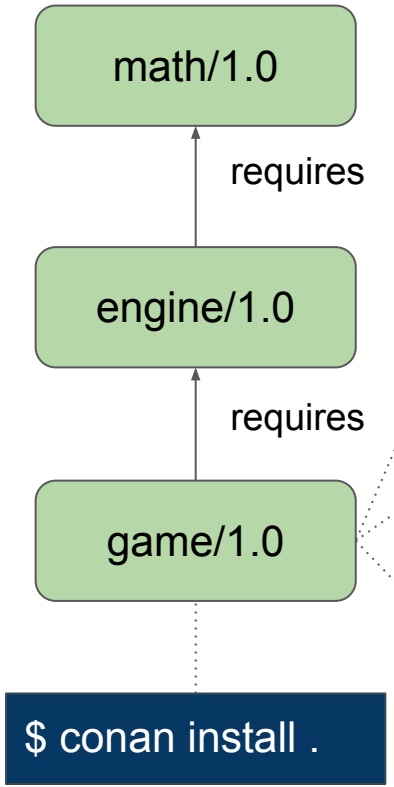
Artifactory servers in production / year



Outline

- The C and C++ native artifacts model
- Conan 1.X packages and dependencies model
- **Limitations of Conan 1.X model**
  - **Transitive headers visibility**
  - Propagation of linkage requirements
  - Dependencies visibility and conflicts
  - Default package_id mode
  - Tool-requires can’t affect package_id
- Conan 2.0 new model
- Demo
- Conclusion

Transitive headers visibility



game/game.cpp

```
#include "engine.h" //OK

int main(){  # game code
}
```

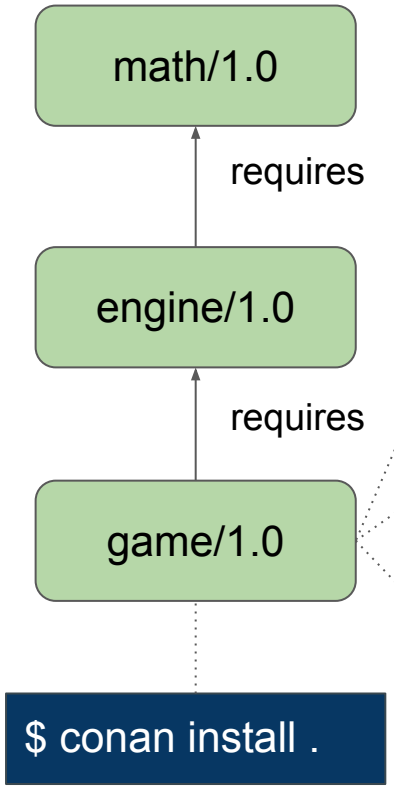
game/conanfile.py

```
from conan import ConanFile
class Game(ConanFile):
    requires = "engine/1.0"
```

game/CMakeLists.txt

```
find_package(engine)
add_executable(game game.cpp)
target_link_libraries(game
                      engine::engine)
```

Transitive headers visibility



game/game.cpp

```
#include "engine.h" //OK
#include "math.h" //SHOULD ERROR
int main(){  # game code
}
```

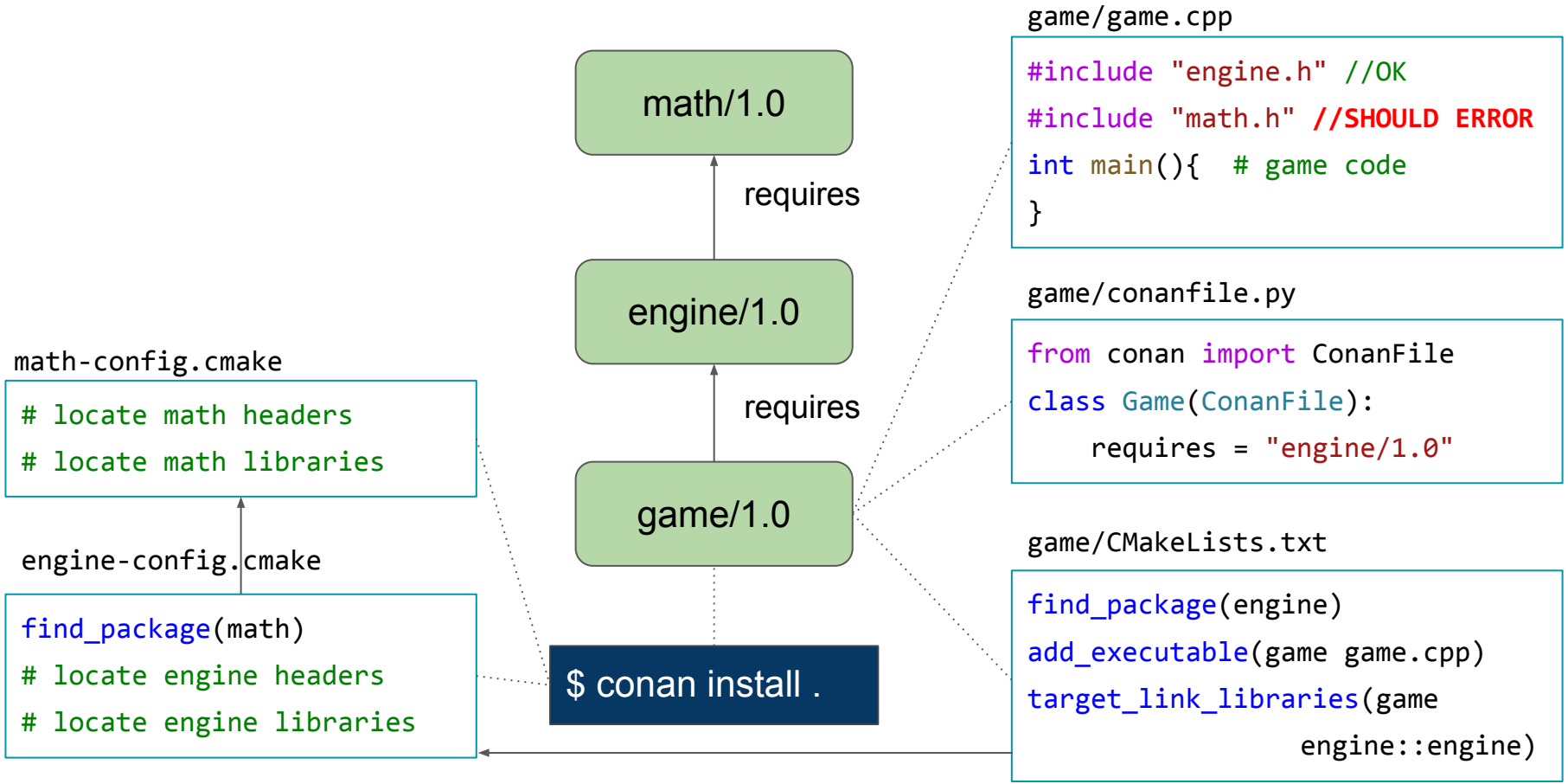
game/conanfile.py

```
from conan import ConanFile
class Game(ConanFile):
    requires = "engine/1.0"
```

game/CMakeLists.txt

```
find_package(engine)
add_executable(game game.cpp)
target_link_libraries(game
                      engine::engine)
```

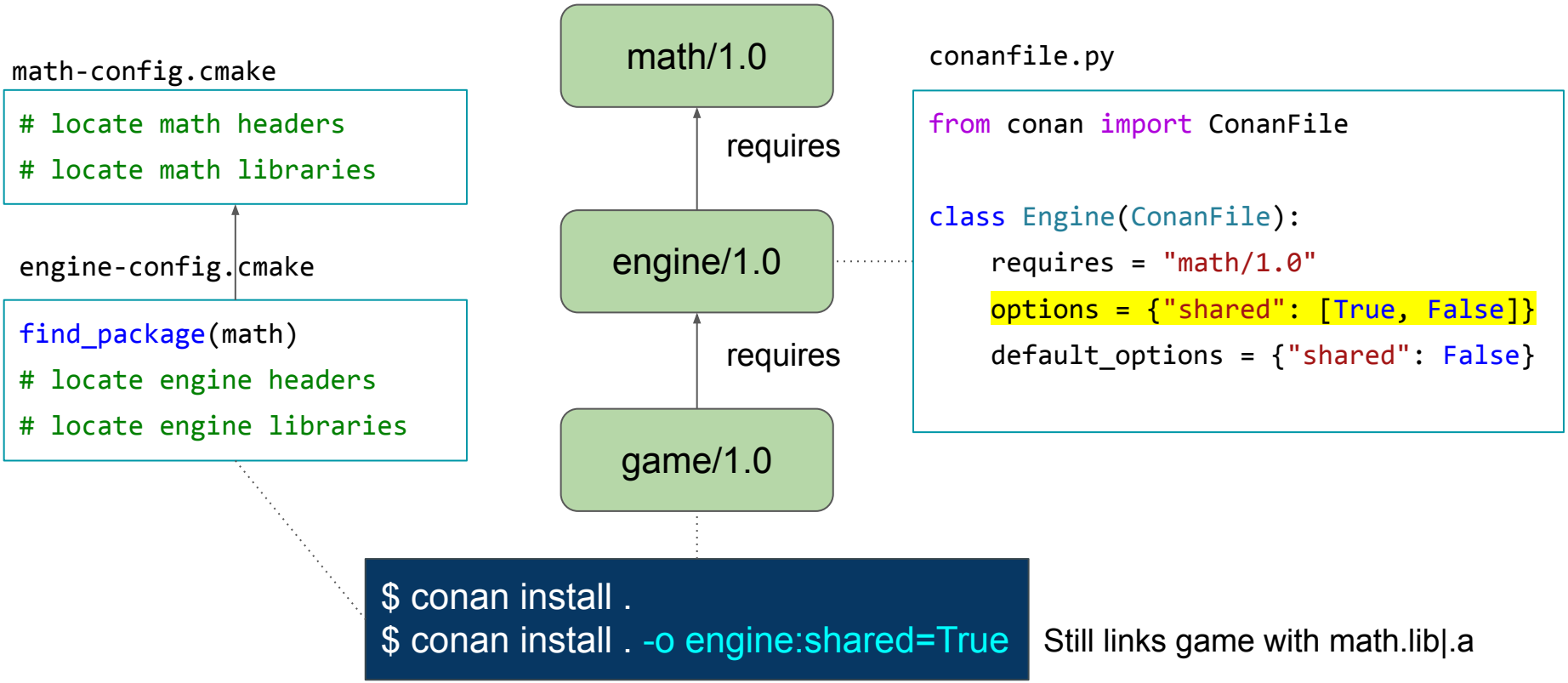
Transitive headers visibility



Outline

- The C and C++ native artifacts model
- Conan 1.X packages and dependencies model
- **Limitations of Conan 1.X model**
  - Transitive headers visibility
  - **Propagation of linkage requirements**
  - Dependencies visibility and conflicts
  - Default package_id mode
  - Tool-requires can’t affect package_id
- Conan 2.0 new model
- Demo
- Conclusion, Q&A

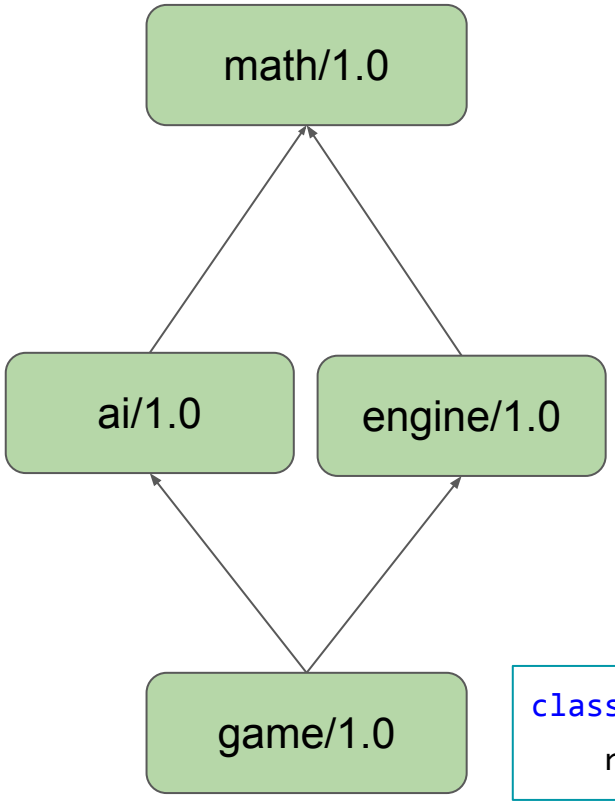
Propagation of linkage



Outline

- The C and C++ native artifacts model
- Conan 1.X packages and dependencies model
- **Limitations of Conan 1.X model**
  - Transitive headers visibility
  - Propagation of linkage requirements
  - **Dependencies visibility and conflicts**
  - Default package_id mode
  - Tool-requires can’t affect package_id
- Conan 2.0 new model
- Demo
- Conclusion, Q&A

Dependency graph: Diamond

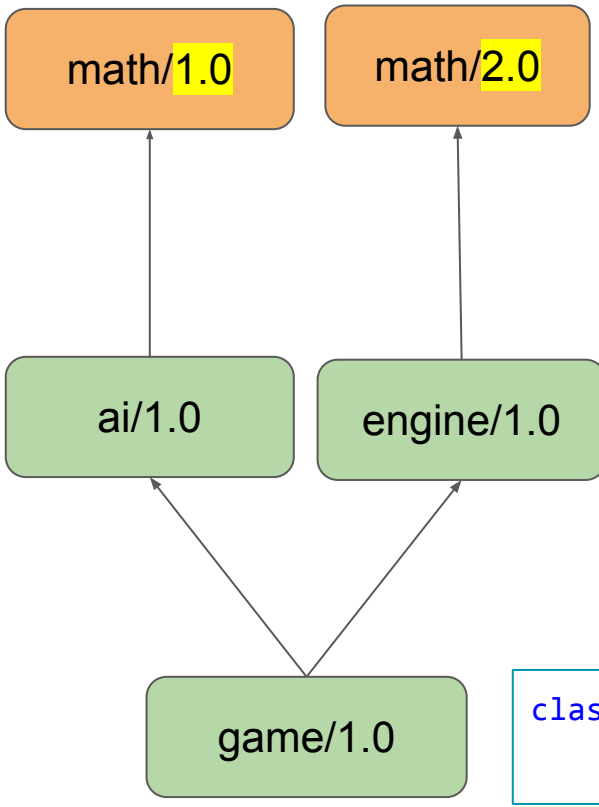



```
class AI(ConanFile):
    requires = "math/1.0"
```

```
class Engine(ConanFile):
    requires = "math/1.0"
```

```
class Game(ConanFile):
    requires = "ai/1.0","engine/1.0"
```

Version conflict



ERROR!

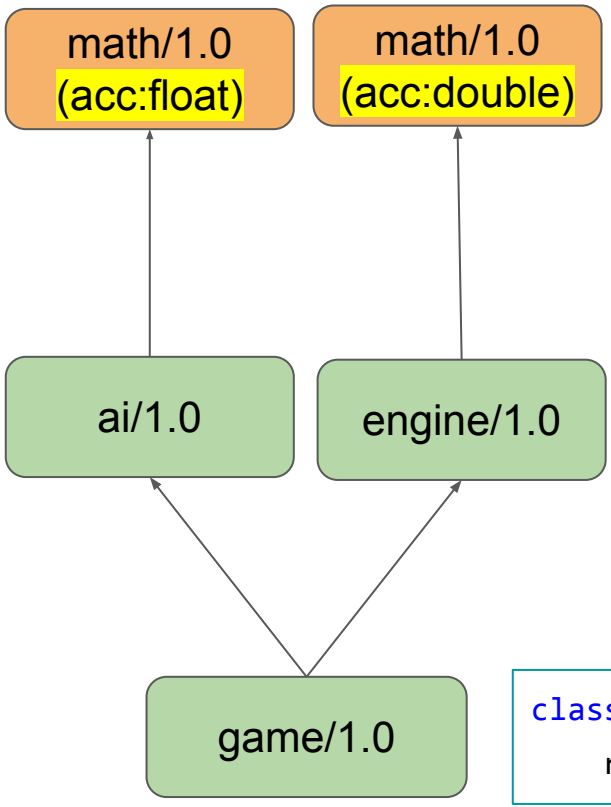
```
class AI(ConanFile):
    requires = "math/1.0"
```

```
class Engine(ConanFile):
    requires = "math/2.0"
```

```
class Game(ConanFile):
    requires = "ai/1.0","engine/1.0"
```

Configuration conflict

```
class Math(ConanFile):
    options = {"acc":
     ["double", "float"]}
```



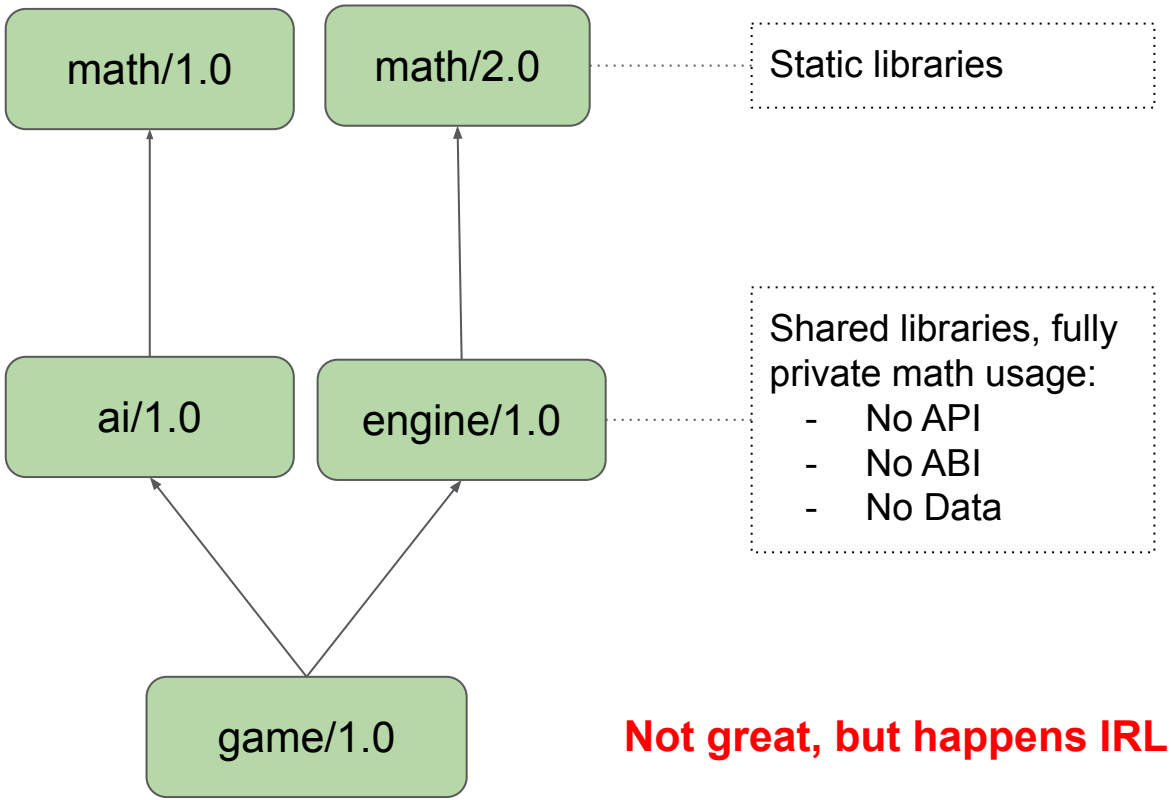
ERROR!

```
class AI(ConanFile):
    requires = "math/1.0"
    default_options =
       {"math:acc": "float"}
```

```
class Engine(ConanFile):
    requires = "math/1.0"
    default_options =
       {"math:acc": "double"}
```

```
class Game(ConanFile):
    requires = "ai/1.0","engine/1.0"
```

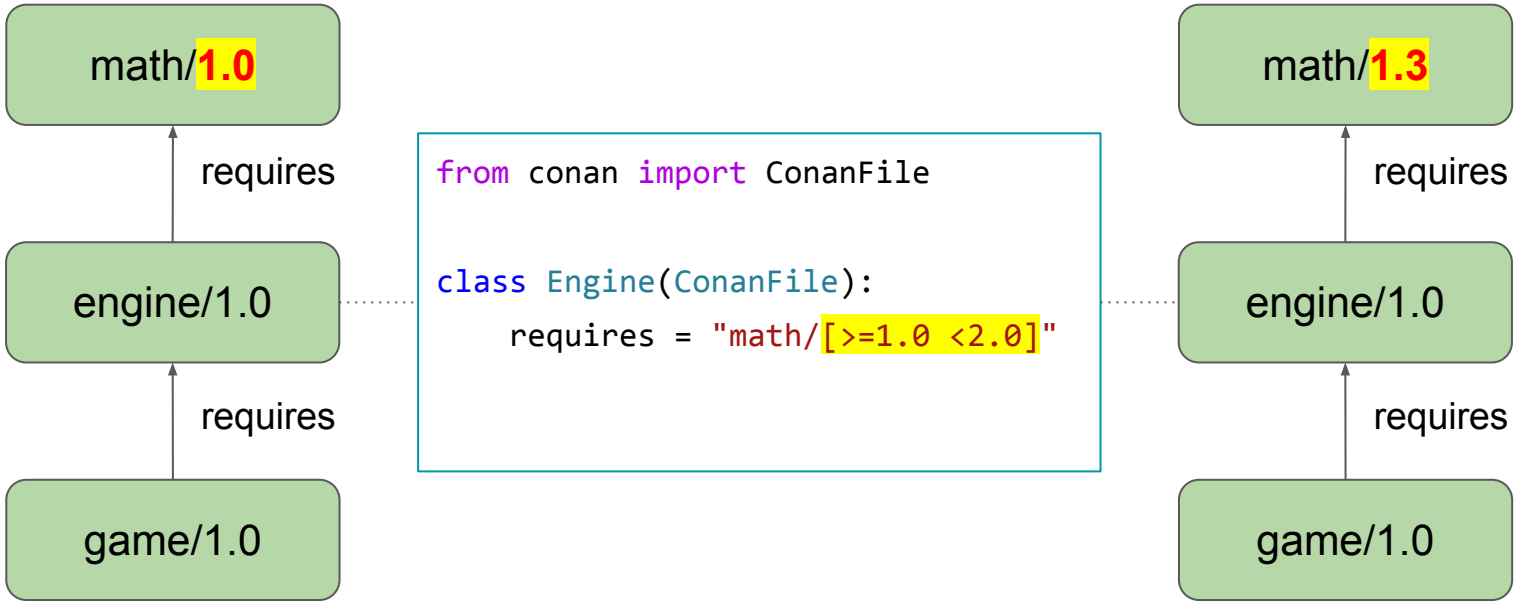
Hidden/private dependencies



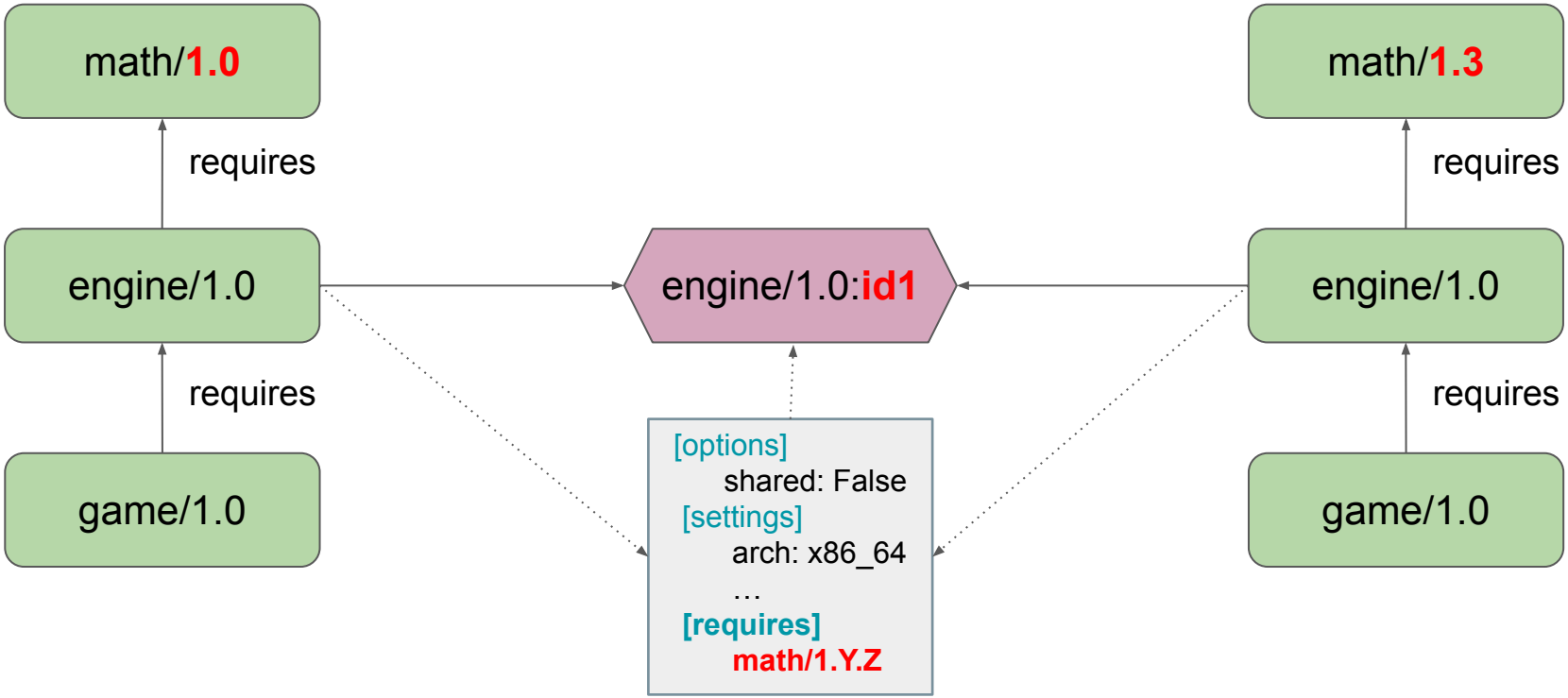
Outline

- The C and C++ native artifacts model
- Conan 1.X packages and dependencies model
- **Limitations of Conan 1.X model**
  - Transitive headers visibility
  - Propagation of linkage requirements
  - Dependencies visibility and conflicts
  - **Default package_id mode**
  - Tool-requires can’t affect package_id
- Conan 2.0 new model
- Demo
- Conclusion, Q&A

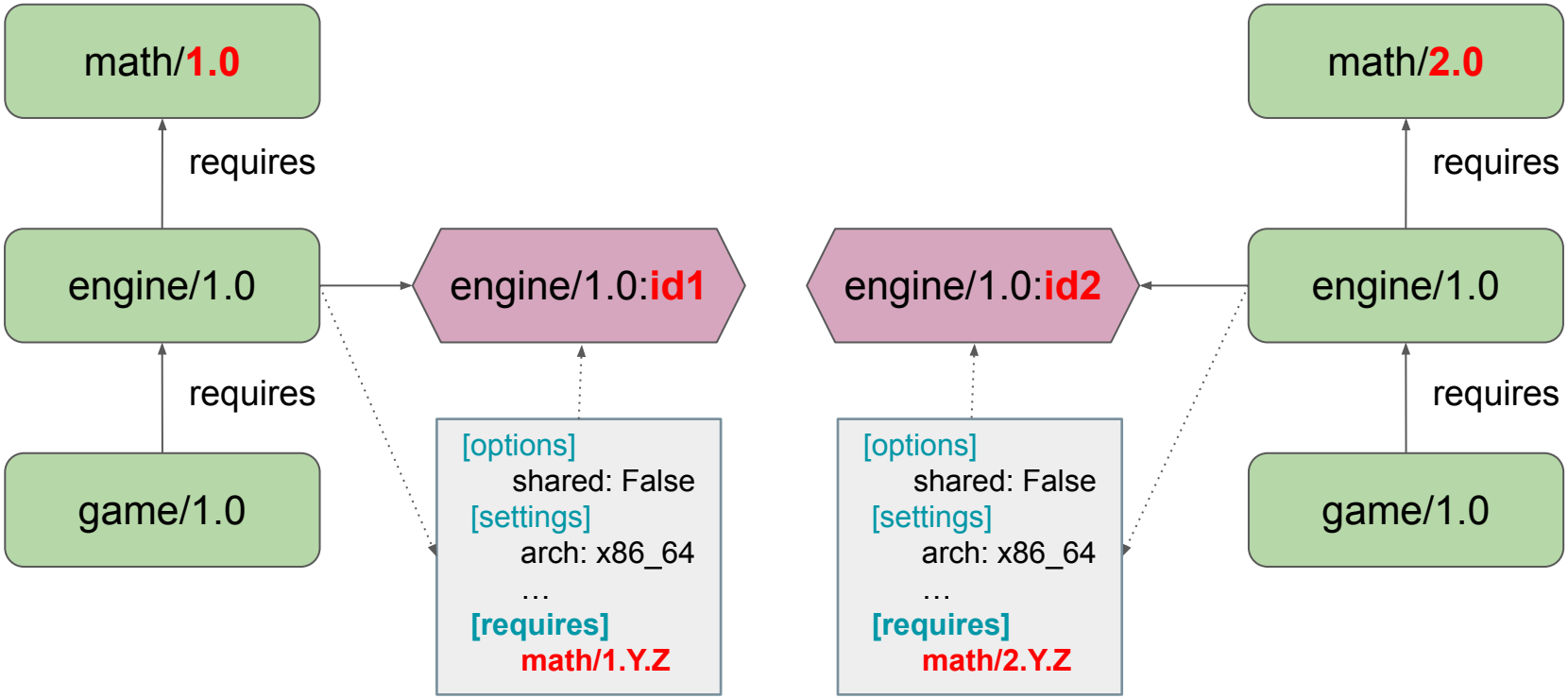
Conan 1.X default package_id_mode = semver



Conan 1.X default package_id_mode = semver



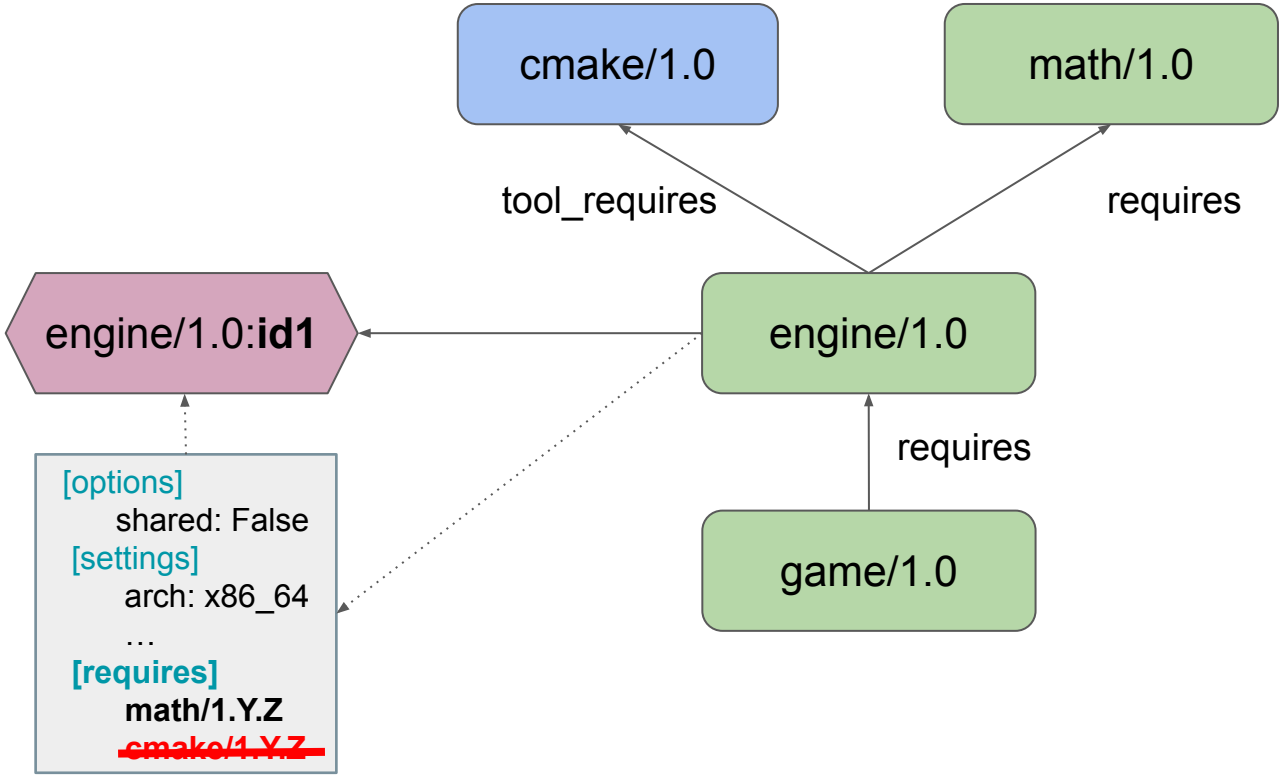
Conan 1.X default package_id_mode = semver



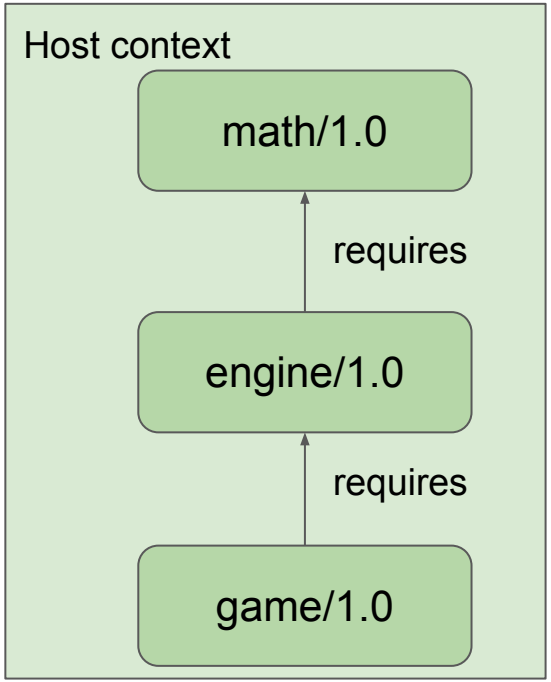
Outline

- The C and C++ native artifacts model
- Conan 1.X packages and dependencies model
- **Limitations of Conan 1.X model**
  - Transitive headers visibility
  - Propagation of linkage requirements
  - Dependencies visibility and conflicts
  - Default package_id mode
  - **Tool-requires can’t affect package_id**
- Conan 2.0 new model
- Demo
- Conclusion, Q&A

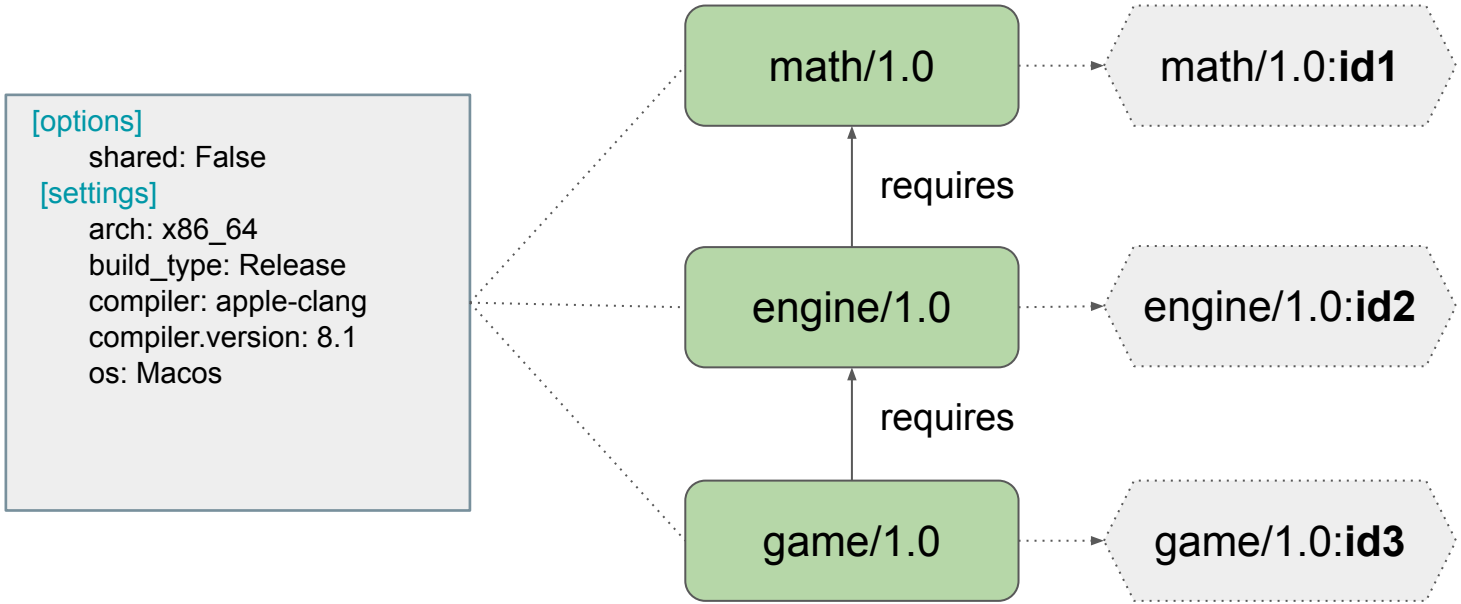
Tool-requires can’t affect package-id



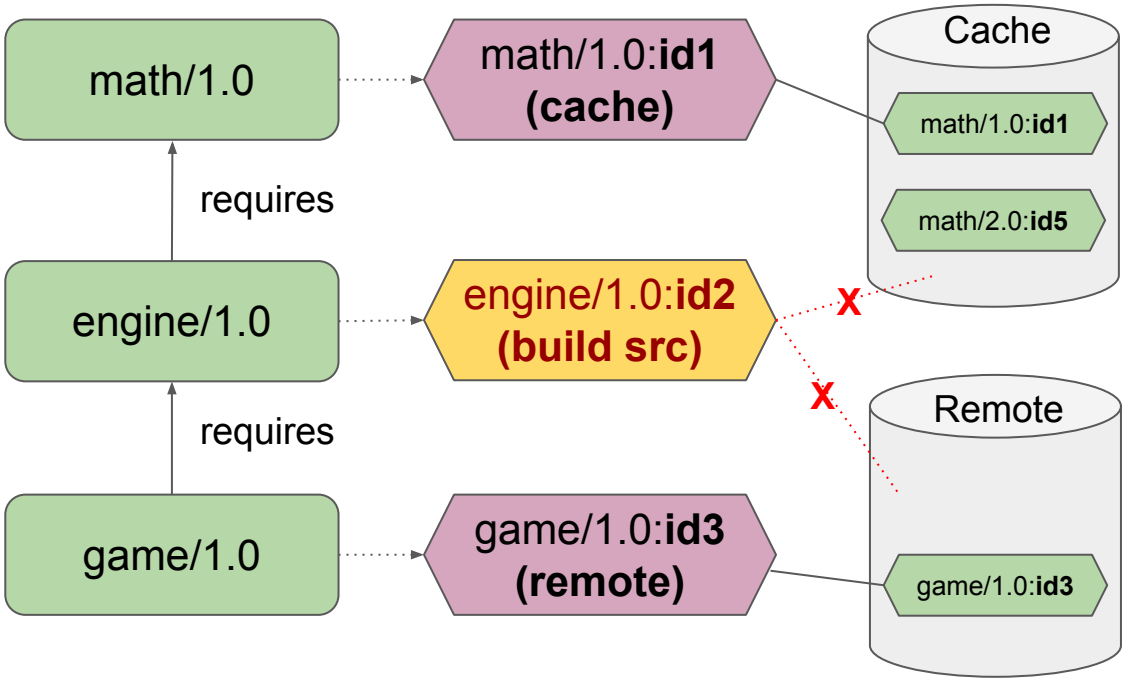
Conan 1.X graph evaluation: Phase I: graph



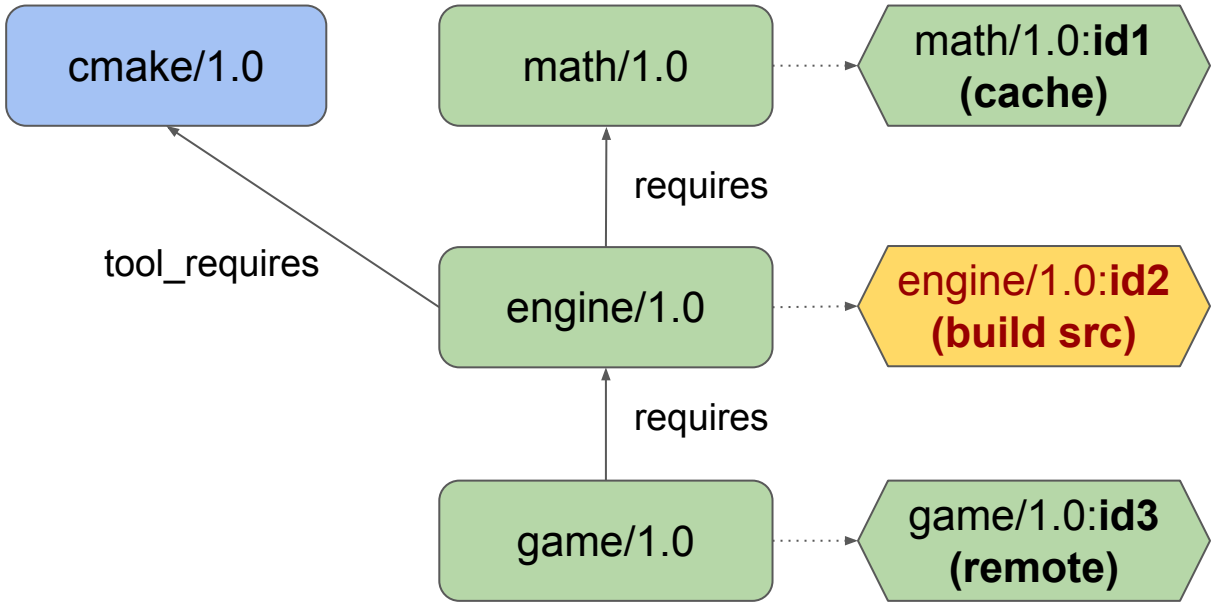
Conan 1.X graph evaluation: Phase II: package-ids



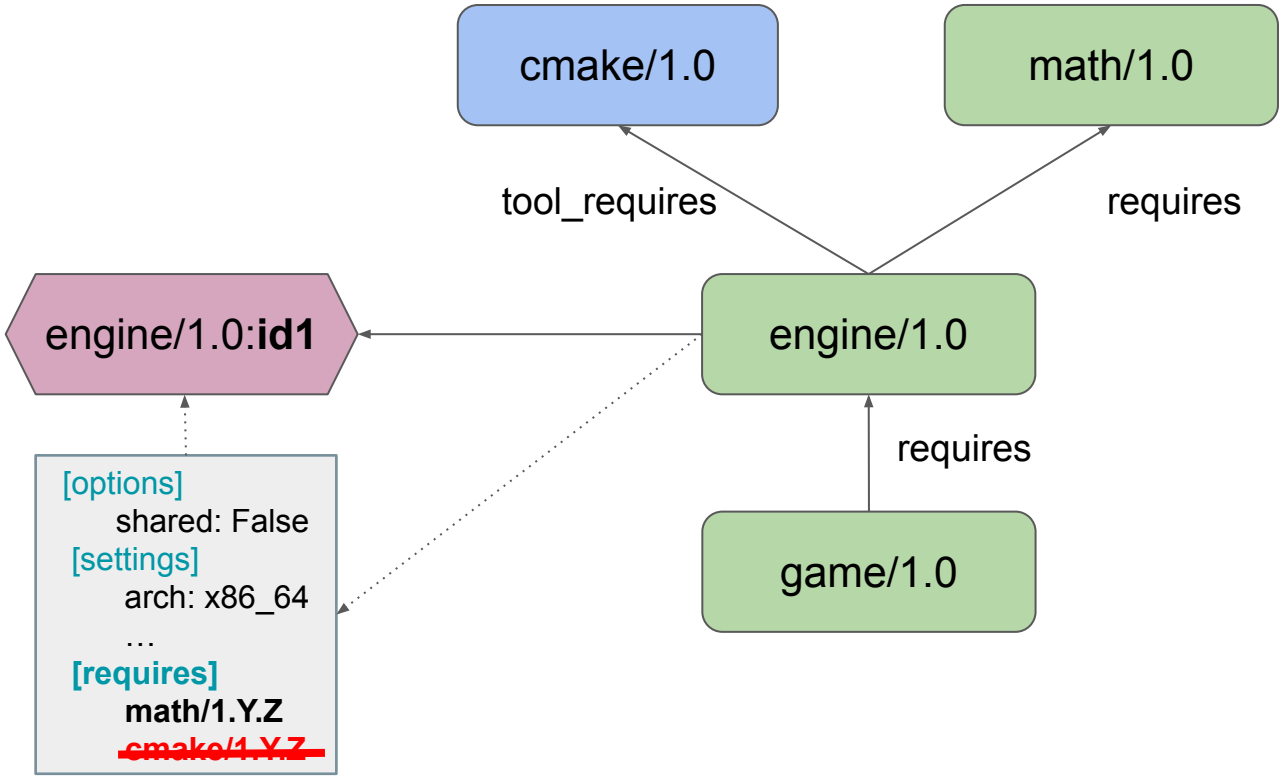
Conan 1.X graph evaluation: Phase III: binaries



Conan 1.X graph evaluation: Phase IV: tool_requires



Tool-requires can’t affect package-id



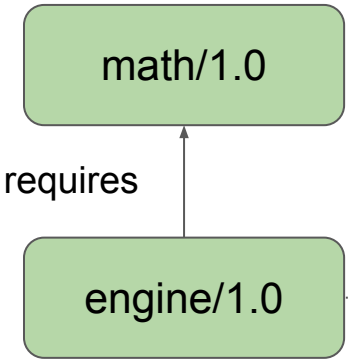
Outline

- The C and C++ native artifacts model
- Conan 1.X packages and dependencies model
- Limitations of Conan 1.X model
- **Conan 2.0 new model**
  - Requirements model
  - Propagation of requirements
  - Headers, libs, run traits
  - Package types
  - Build, visible traits
  - New package_id
- Demo
- Conclusion, Q&A

Conan 2.0

- **New dependency graph model**
  - **New package_id**
- New user custom commands system
- New public Python API
- New binary compatibility approach
- New build system integrations
- New environment management
- New configuration system
- New deployment mechanisms
- New lockfiles and Continuous Integration tools
- New multi-revision cache
- New editable packages and layouts
- & many more…

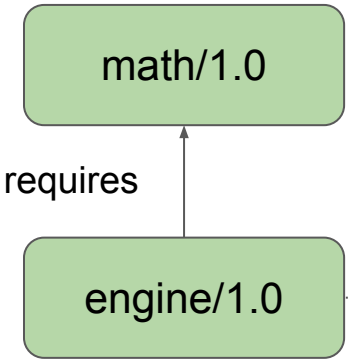
Conan 2.0 proposal



engine/conanfile.py

```
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"

    def requirements(self):
        self.requires("math/1.0")
```

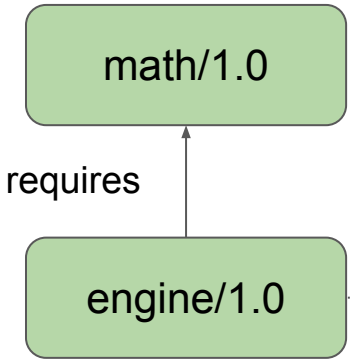
Conan 2.0 proposal: Requirement traits



engine/conanfile.py

```
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"

    def requirements(self):
        self.requires("math/1.0",
                      headers=True, libs=True)
```

Conan 2.0 proposal



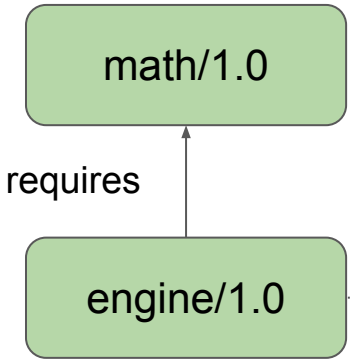
engine/conanfile.py

```python
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"

    def requirements(self):
        self.requires("math/1.0",
                      headers=True, libs=True)
```

math-config.cmake

```
set_property(TARGET math::math PROPERTY INTERFACE_LINK_LIBRARIES ...)
set_property(TARGET math::math PROPERTY INTERFACE_INCLUDE_DIRECTORIES ...)
```

Conan 2.0 proposal



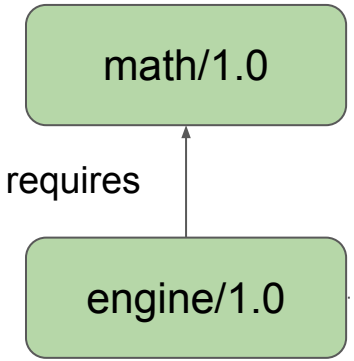
engine/conanfile.py

```python
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"

    def requirements(self):
        self.requires("math/1.0",
                      headers=False, libs=True)
```

math-config.cmake

```
set_property(TARGET math::math PROPERTY INTERFACE_LINK_LIBRARIES ...)
set_property(TARGET math::math PROPERTY INTERFACE_INCLUDE_DIRECTORIES ...)
```

Conan 2.0 proposal



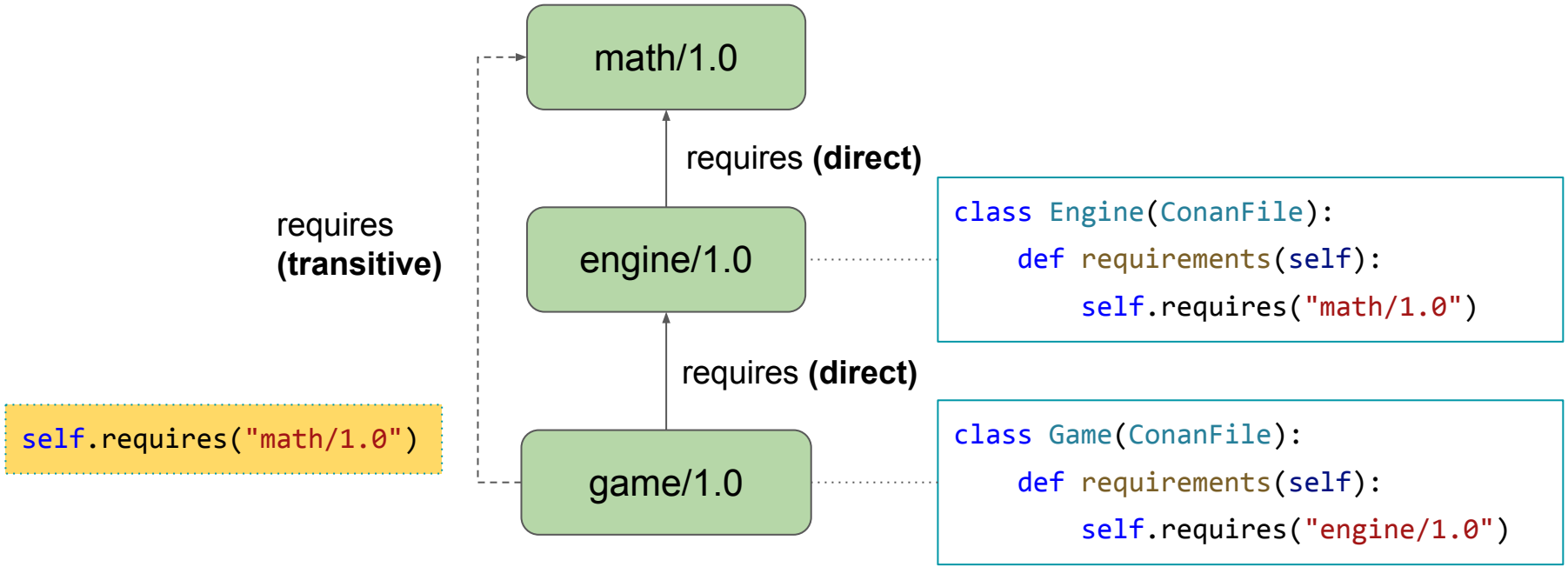
engine/conanfile.py

```python
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"

    def requirements(self):
        self.requires("math/1.0",
                      headers=True, libs=False)
```

math-config.cmake

```
set_property(TARGET math::math PROPERTY INTERFACE_LINK_LIBRARIES ...)
set_property(TARGET math::math PROPERTY INTERFACE_INCLUDE_DIRECTORIES ...)
```

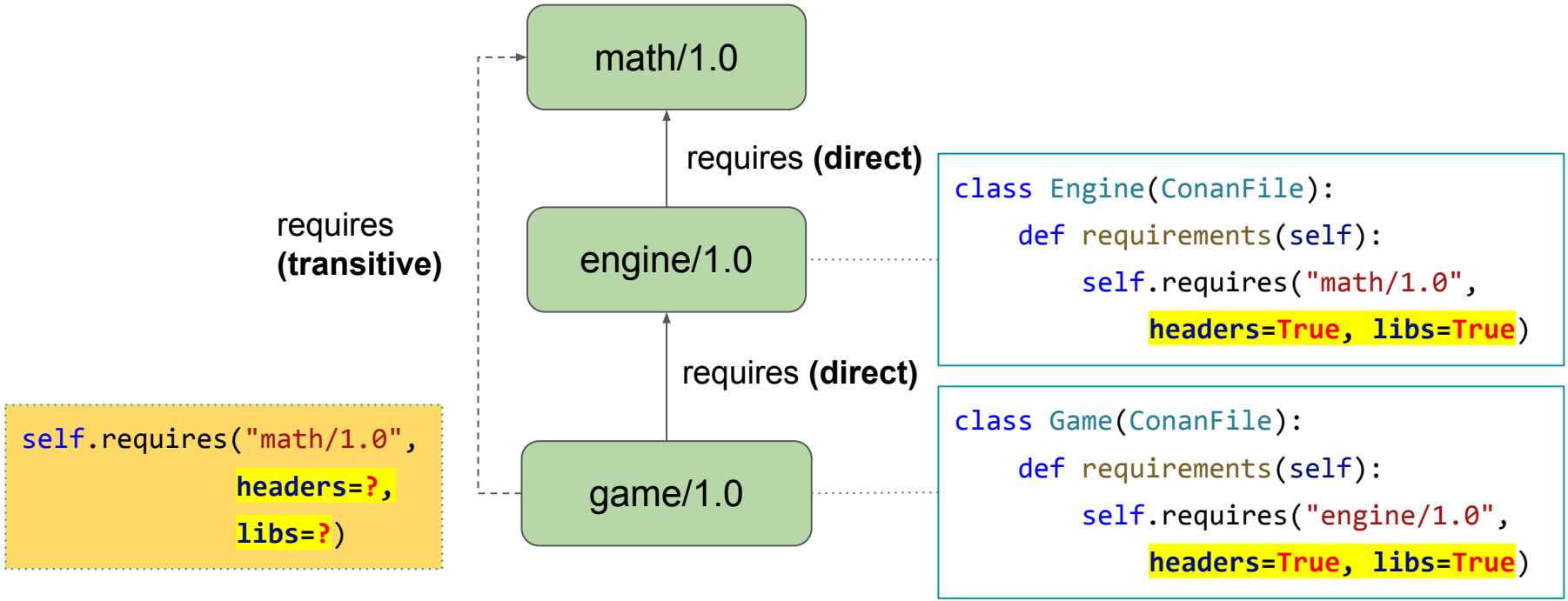
Conan 2.0 proposal: Direct vs. transitive dependencies



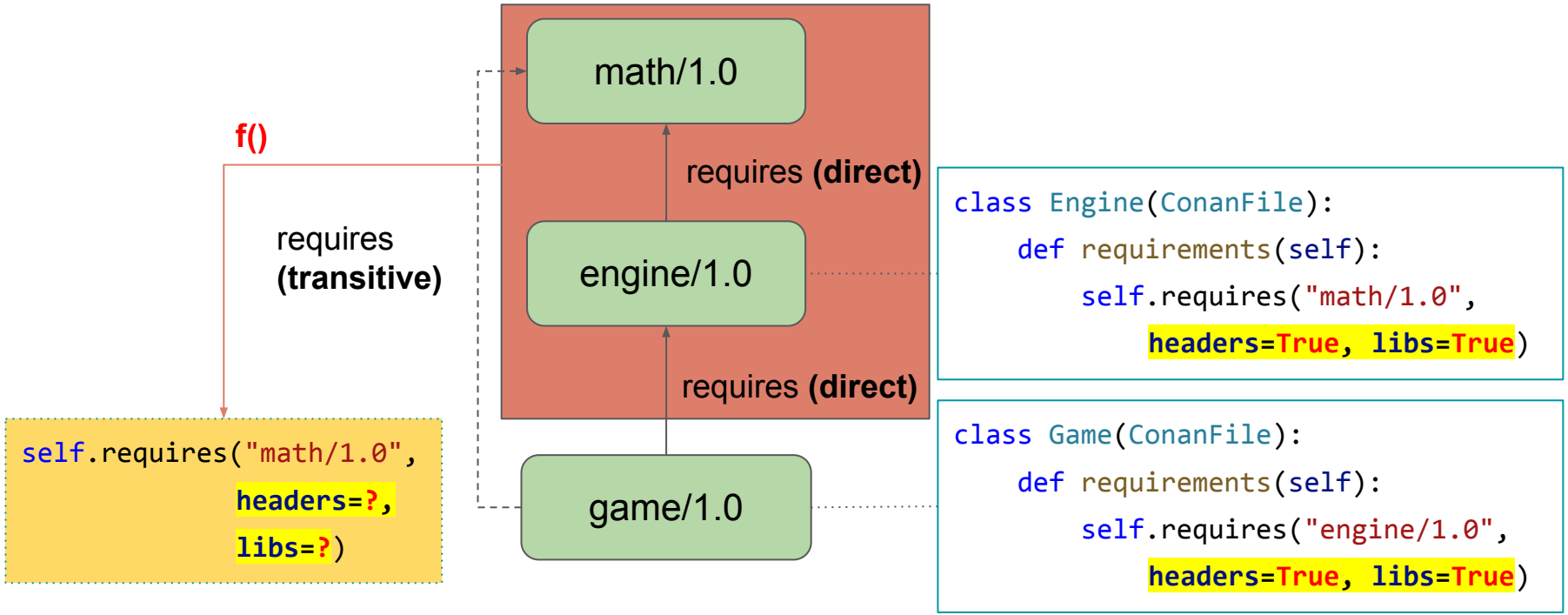
Outline

- The C and C++ native artifacts model
- Conan 1.X packages and dependencies model
- Limitations of Conan 1.X model
- **Conan 2.0 new model**
  - Requirements model
  - **Propagation of requirements**
  - Headers, libs, run traits
  - Package types
  - Build, visible traits
  - New package_id
- Demo
- Conclusion, Q&A

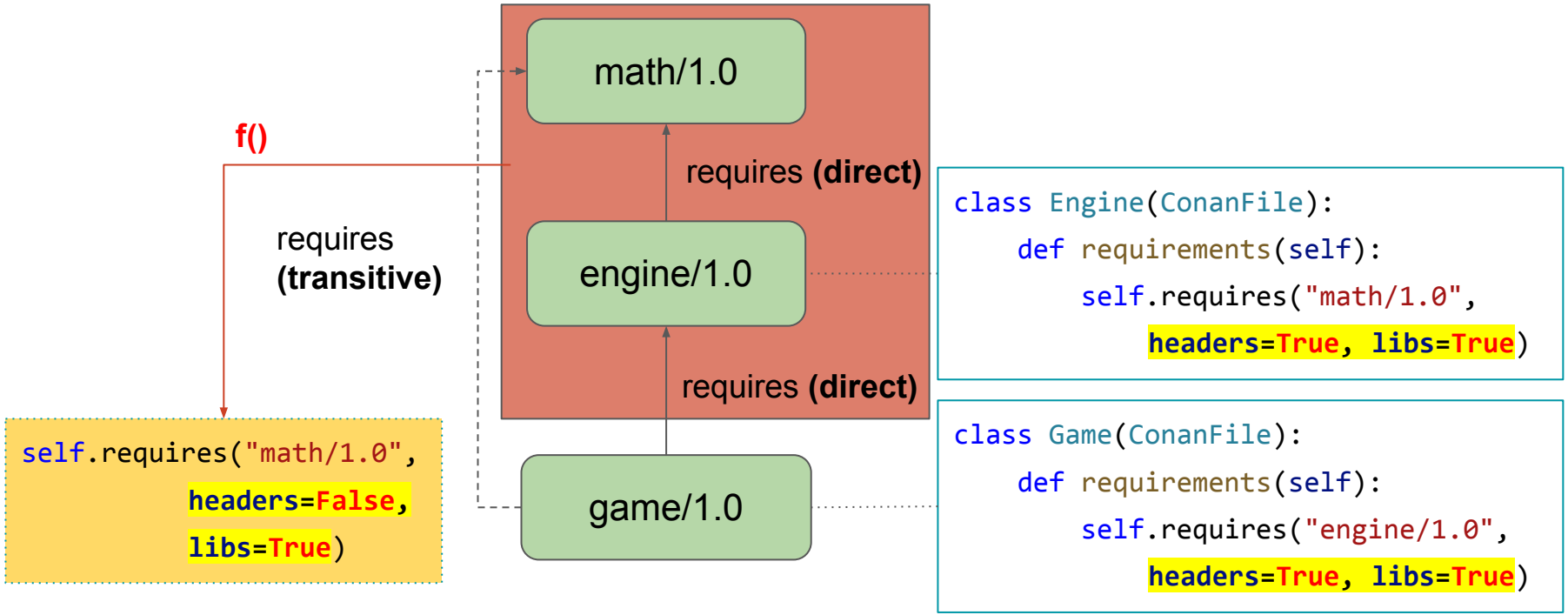
Conan 2.0 proposal: Propagation of Requirement traits



Conan 2.0 proposal: Propagation of Requirement traits



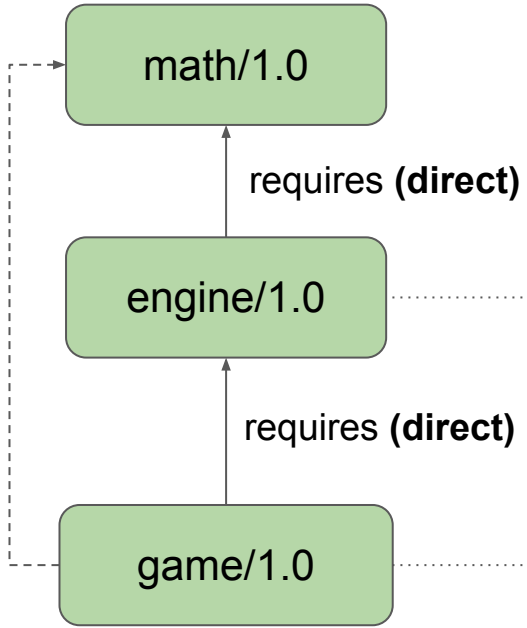
Conan 2.0 proposal: Propagation of Requirement traits



Outline

- The C and C++ native artifacts model
- Conan 1.X packages and dependencies model
- Limitations of Conan 1.X model
- **Conan 2.0 new model**
  - Requirements model
  - Propagation of requirements
  - **Headers, libs, run traits**
  - Package types
  - Build, visible traits
  - New package_id
- Demo
- Conclusion

Transitive headers no longer propagated



requires **(transitive)**

```
self.requires("math/1.0",
              headers=False,
              libs=True)
```

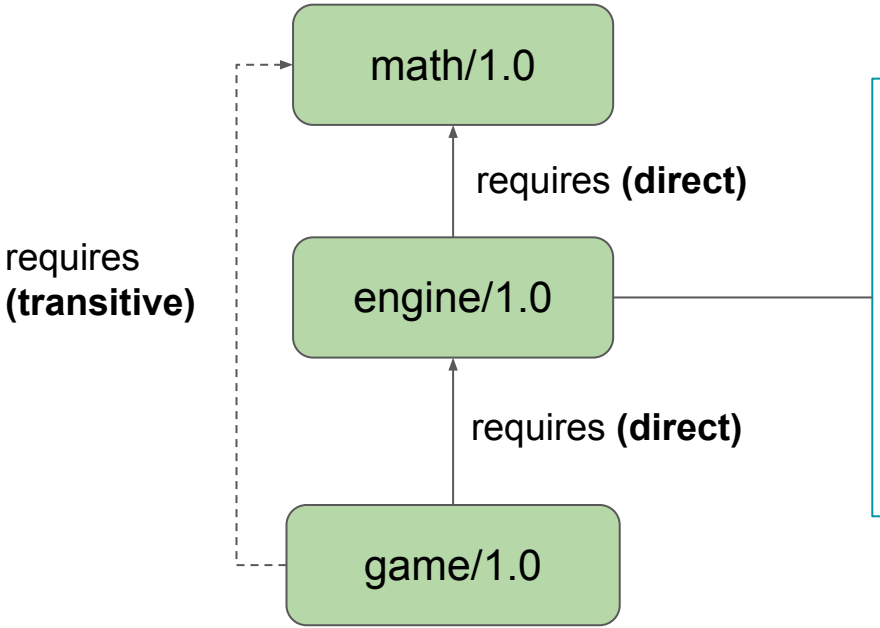
```
class Engine(ConanFile):
    def requirements(self):
        self.requires("math/1.0",
                      headers=True, libs=True)
```

```
class Game(ConanFile):
    def requirements(self):
        self.requires("engine/1.0",
                      headers=True, libs=True)
```

math-config.cmake

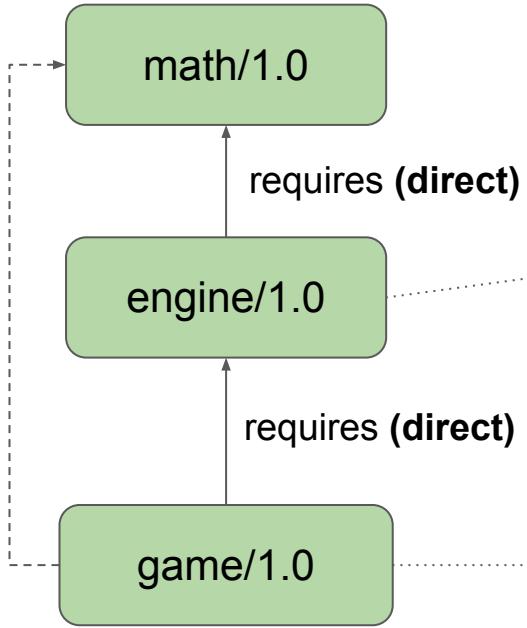
```
set_property(TARGET math::math PROPERTY INTERFACE_LINK_LIBRARIES ...)
set_property(TARGET math::math PROPERTY INTERFACE_INCLUDE_DIRECTORIES ...)
```

Headers propagated via public headers




```
#include "math/matrix.h"

class Engine {
    // ...
    void check_move(
         const matrix& m1);
};
```

Headers propagated via public headers



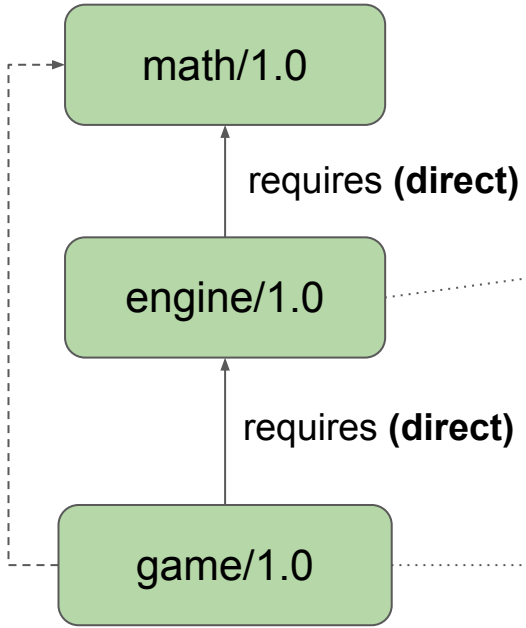
requires **(transitive)**

```
self.requires("math/1.0",
              headers=True,
              libs=True)
```

```
class Engine(ConanFile):
    def requirements(self):
        self.requires("math/1.0",
                      headers=True, libs=True,
                      transitive_headers=True)
```

```
class Game(ConanFile):
    def requirements(self):
        self.requires("engine/1.0",
                      headers=True, libs=True)
```

Libraries hidden (engine shared, math static)



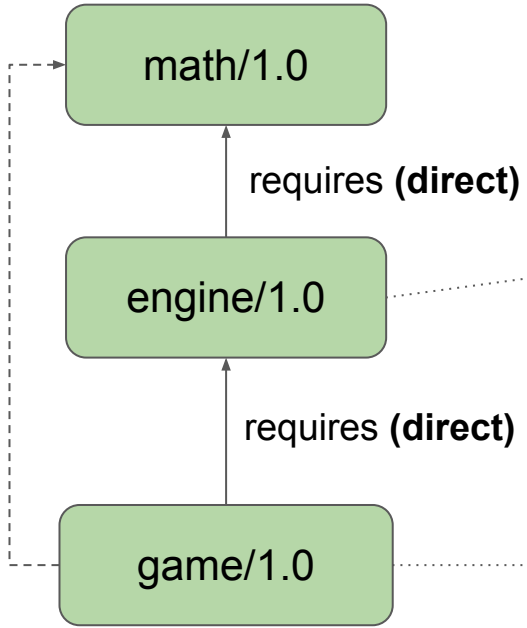
requires **(transitive)**

```
self.requires("math/1.0",
              headers=False,
              libs=False)
```

```
class Engine(ConanFile):
    def requirements(self):
        self.requires("math/1.0",
                      headers=True, libs=True,
                      transitive_libs=False)
```

```
class Game(ConanFile):
    def requirements(self):
        self.requires("engine/1.0",
                      headers=True, libs=True)
```

Hiding embedded libraries, solved




```
class Engine(ConanFile):
    def requirements(self):
        self.requires("math/1.0",
                      headers=True, libs=True,
                      transitive_libs=False)
```

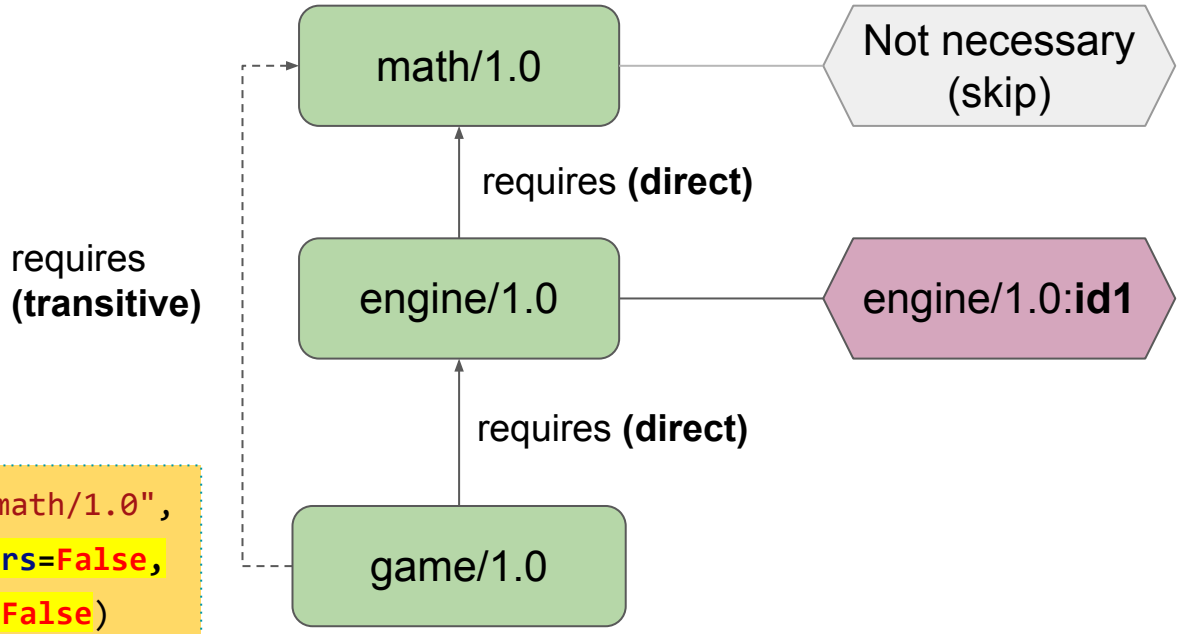
```
class Game(ConanFile):
    def requirements(self):
        self.requires("engine/1.0",
                      headers=True, libs=True)
```

```
self.requires("math/1.0",
              headers=False,
              libs=False)
```

math-config.cmake

```
set_property(TARGET math::math PROPERTY INTERFACE_LINK_LIBRARIES ...)
set_property(TARGET math::math PROPERTY INTERFACE_INCLUDE_DIRECTORIES ...)
```

Hiding embedded libraries, + efficient

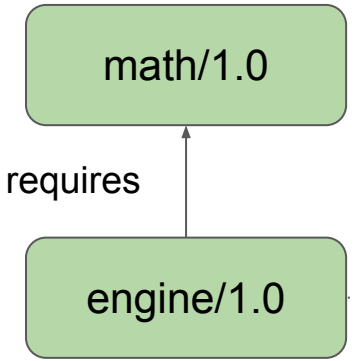



```
self.requires("math/1.0",
              headers=False,
              libs=False)
```

math-config.cmake

```
set_property(TARGET math::math PROPERTY INTERFACE_LINK_LIBRARIES ...)
set_property(TARGET math::math PROPERTY INTERFACE_INCLUDE_DIRECTORIES ...)
```

Runtime trait



conanfile.py

```
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"

    def requirements(self):
        self.requires("math/1.0",
           headers=True, libs=True, run=True)
```

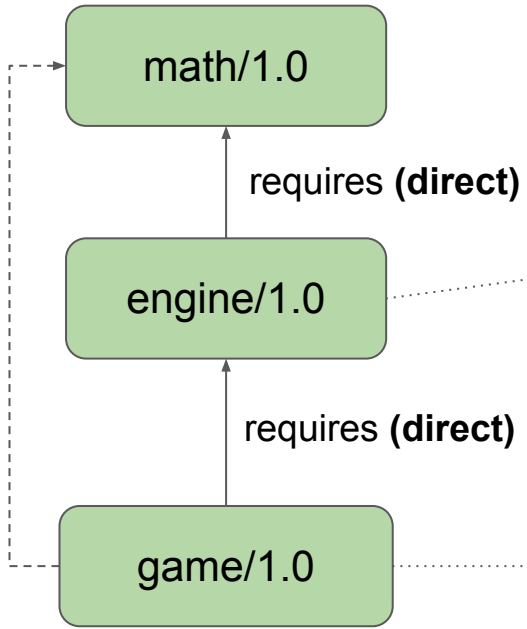
math-config.cmake

```
set_property(TARGET math::math ...)
```

conanrun.sh (or .bat)

```
export PATH=<path/to/math>/bin:$PATH
```

Runtime trait



requires
(transitive)

```
self.requires("math/1.0",
              headers=False,
              libs=False,
              run=True)
```

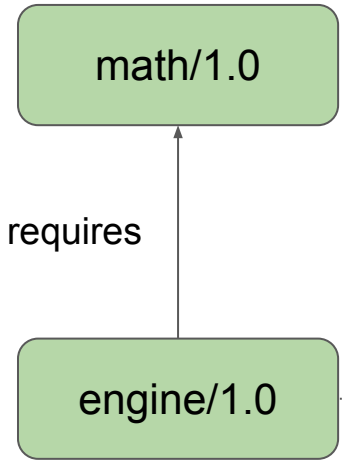
```
class Engine(ConanFile):
    def requirements(self):
        self.requires("math/1.0",
                      headers=True, libs=True,
                      run=True)
```

```
class Game(ConanFile):
    def requirements(self):
        self.requires("engine/1.0",
                      headers=True, libs=True)
```

Outline

- The C and C++ native artifacts model
- Conan 1.X packages and dependencies model
- Limitations of Conan 1.X model
- **Conan 2.0 new model**
  - Requirements model
  - Propagation of requirements
  - Headers, libs, run traits
  - **Package types**
  - Build, visible, override/force traits
  - New package_id
- Demo
- Conclusion, Q&A

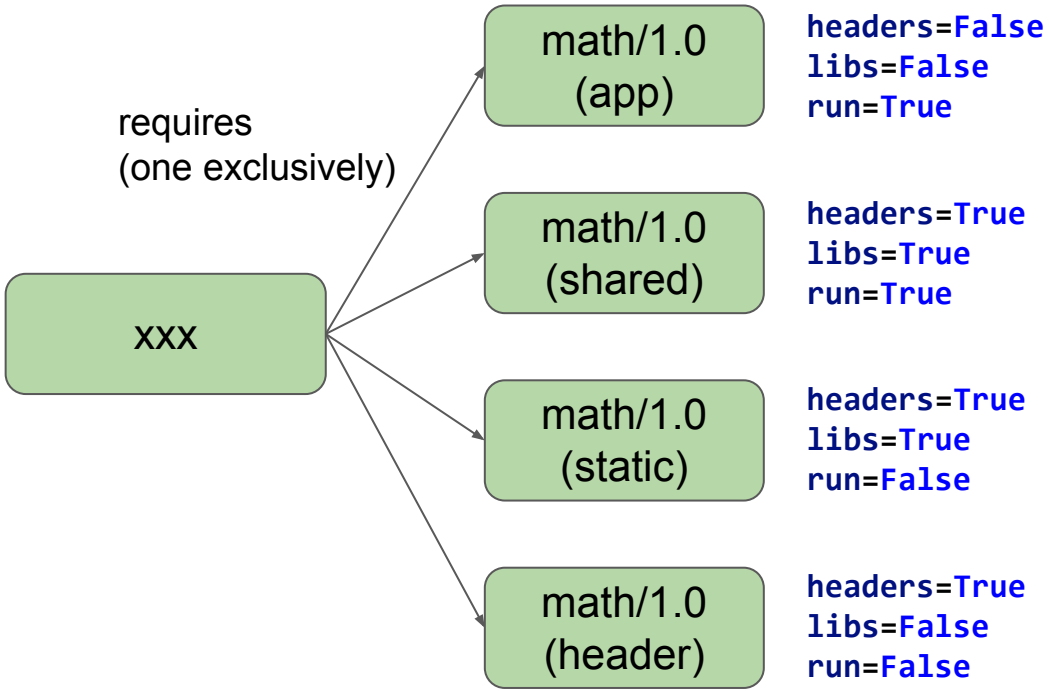
Recap: traits



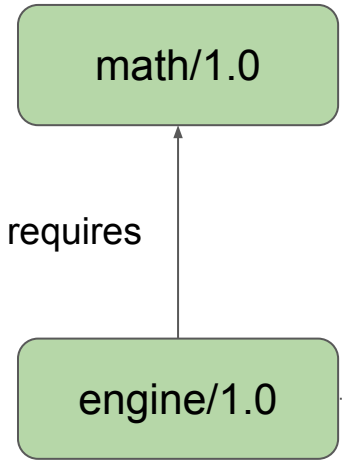
engine/conanfile.py

```
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"

    def requirements(self):
        self.requires("math/1.0",
                      headers=True,
                      libs=True,
                      run=False,
                      transitive_headers=False,
                      transitive_libs=False)
```

Recap: traits typical values



Recap: traits



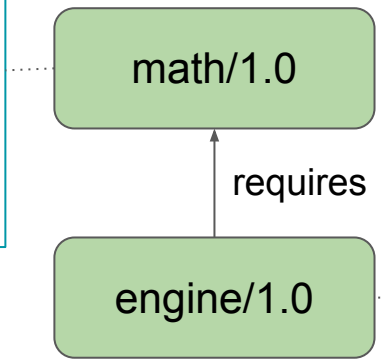
engine/conanfile.py

```
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"

    def requirements(self):
        self.requires("math/1.0",
                      headers=?,
                      libs=?,
                      run=?,
                      transitive_headers=?,
                      transitive_libs=?)
```

Welcome “package_type”

math/conanfile.py

```
from conan import ConanFile

class Math(ConanFile):
    name = "math"
    version = "1.0"
    package_type= "static-library"
```



engine/conanfile.py

```
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"
    requires = "math/1.0"
    package_type= "static-library"
```

Deducing “package_type”

math/conanfile.py

```
from conan import ConanFile


class Math(ConanFile):
    name = "math"
    version = "1.0"
    options = {"shared": [True, False],
               "header_only": [True, False]}
```

math/1.0

shared=True ⇒ package_type = “shared-library”
shared=False ⇒ package_type = “static-library”
header_only=True ⇒ package_type = “header-library”

Known “package_type”s

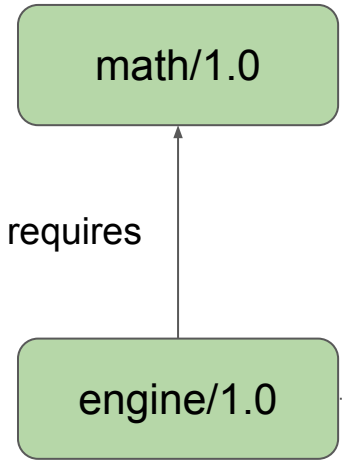
math/conanfile.py

```
from conan import ConanFile

class Math(ConanFile):
    name = "math"
    version = "1.0"
    package_type= "?"
```

math/1.0

- application
- shared-library
- static-library
- header-library
- library
- build-scripts
- python-require
- unknown

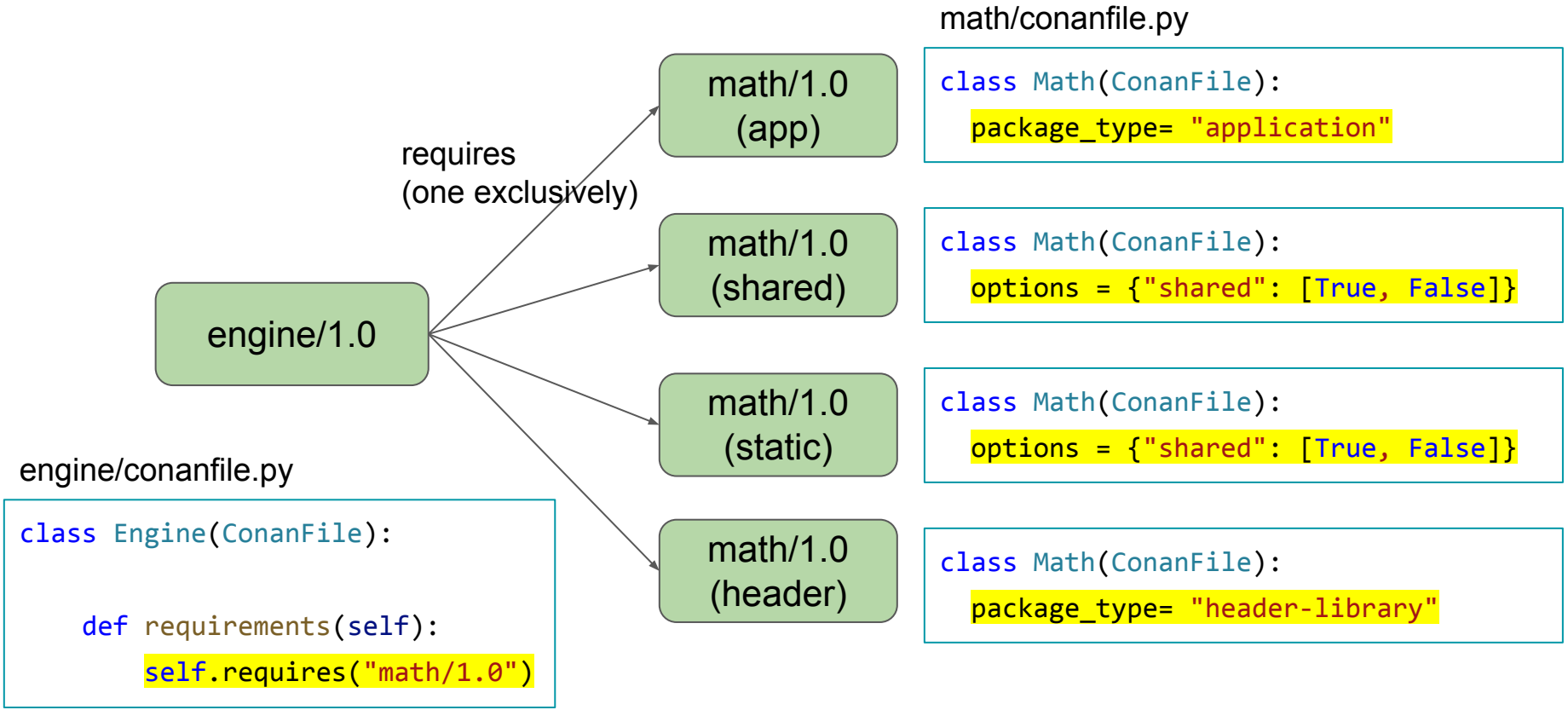
Traits defaults



engine/conanfile.py

```
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"

    def requirements(self):
        self.requires("math/1.0",
                      headers=True,
                      libs=True,
                      run=None,
                      transitive_headers=None,
                      transitive_libs=None)
```

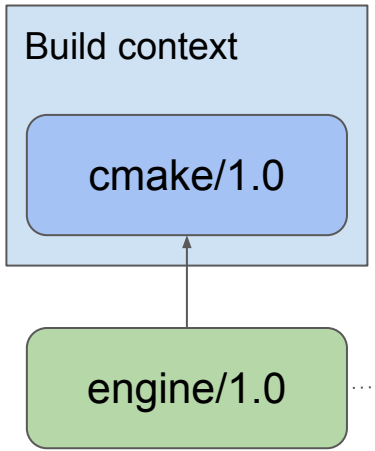
Recap: traits typical values



Outline

- The C and C++ native artifacts model
- Conan 1.X packages and dependencies model
- Limitations of Conan 1.X model
- **Conan 2.0 new model**
  - Requirements model
  - Propagation of requirements
  - Headers, libs, run traits
  - Package types
  - **Build, visible, override/force traits**
  - New package_id
- Demo
- Conclusion

Build trait



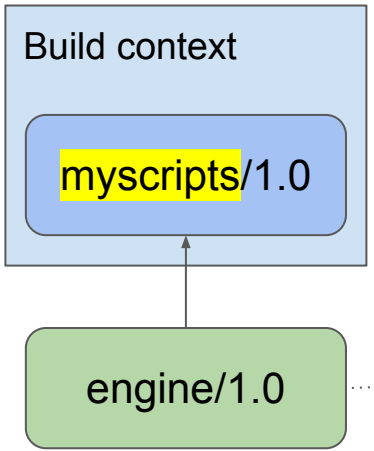
engine/conanfile.py

```
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"

    def requirements(self):
        self.requires("cmake/1.0",
           headers=False, libs=False,
           run=True, build=True)
```

conanbuild.sh (or .bat)

```
export PATH=<path/to/cmake>/bin:$PATH
```

Build trait

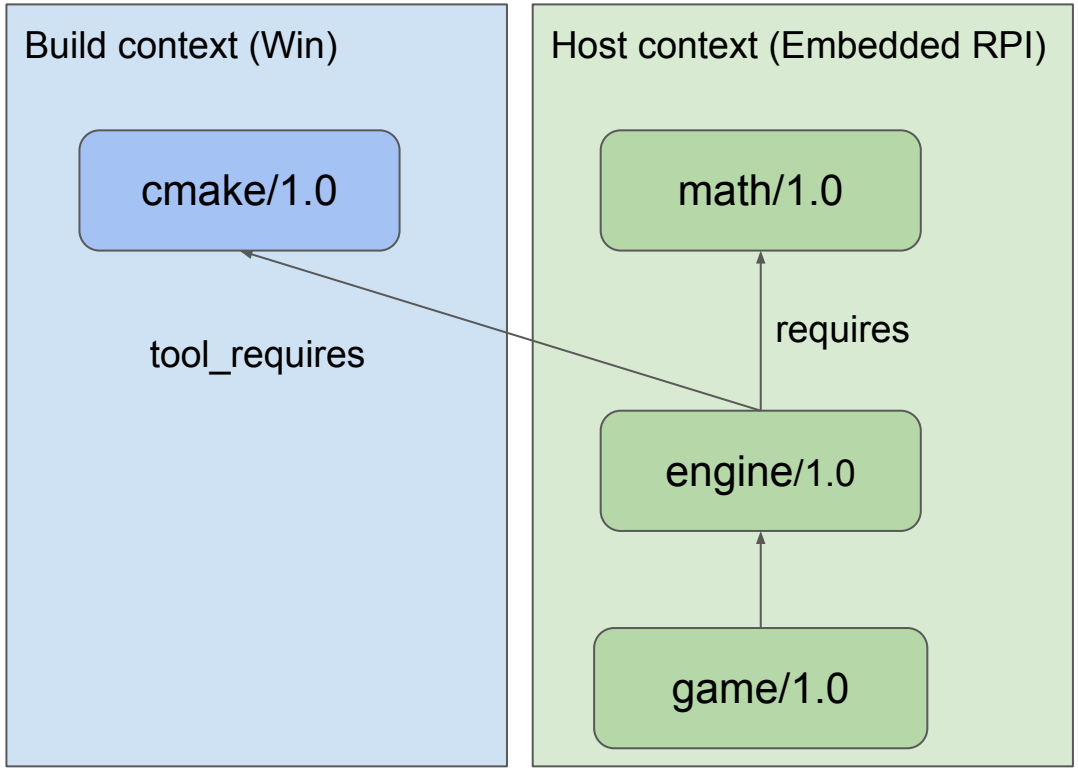


engine/conanfile.py

```
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"

    def requirements(self):
        self.requires("myscripts/1.0",
           headers=False, libs=False,
           run=False, build=True)
```

```
No file generated. Used via
self.dependencies["myscripts"].package_folder
```

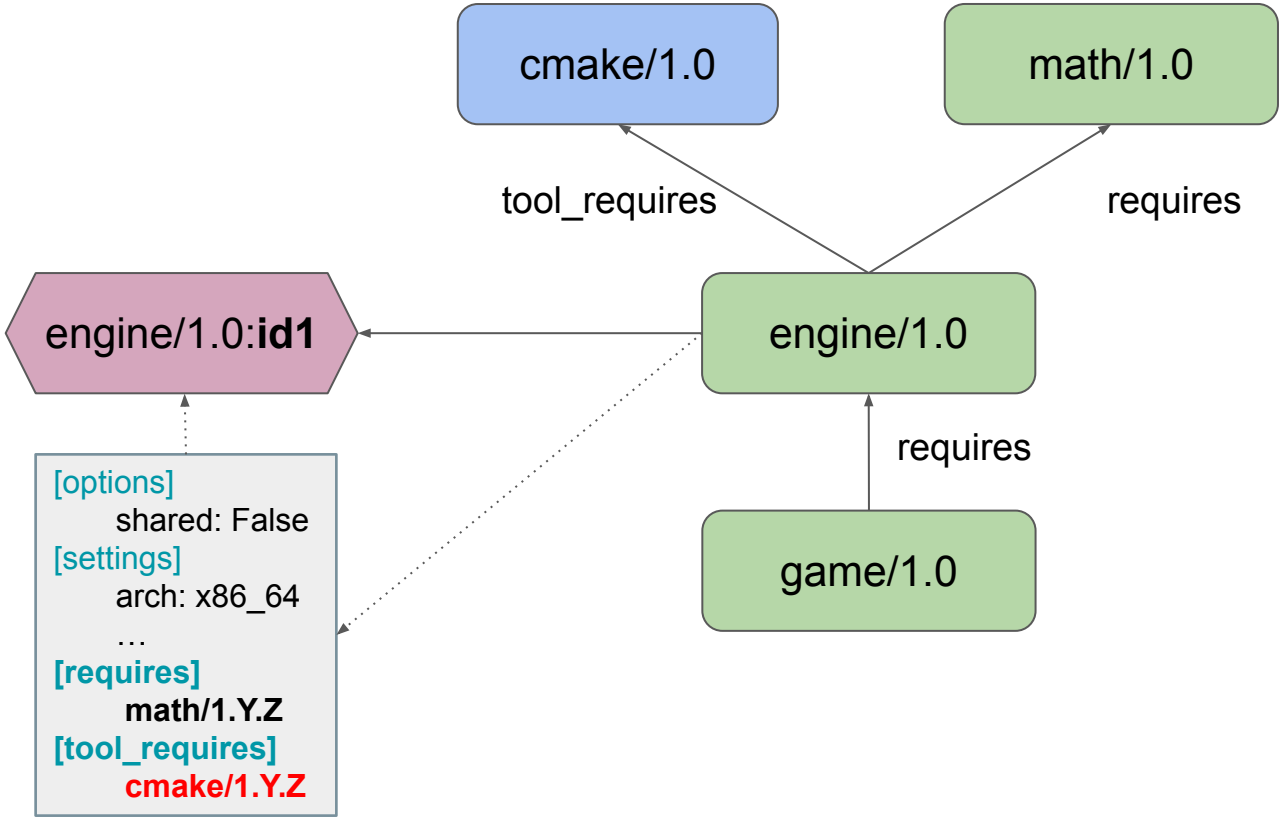
Conan 2.X graph evaluation: Phase I: graph



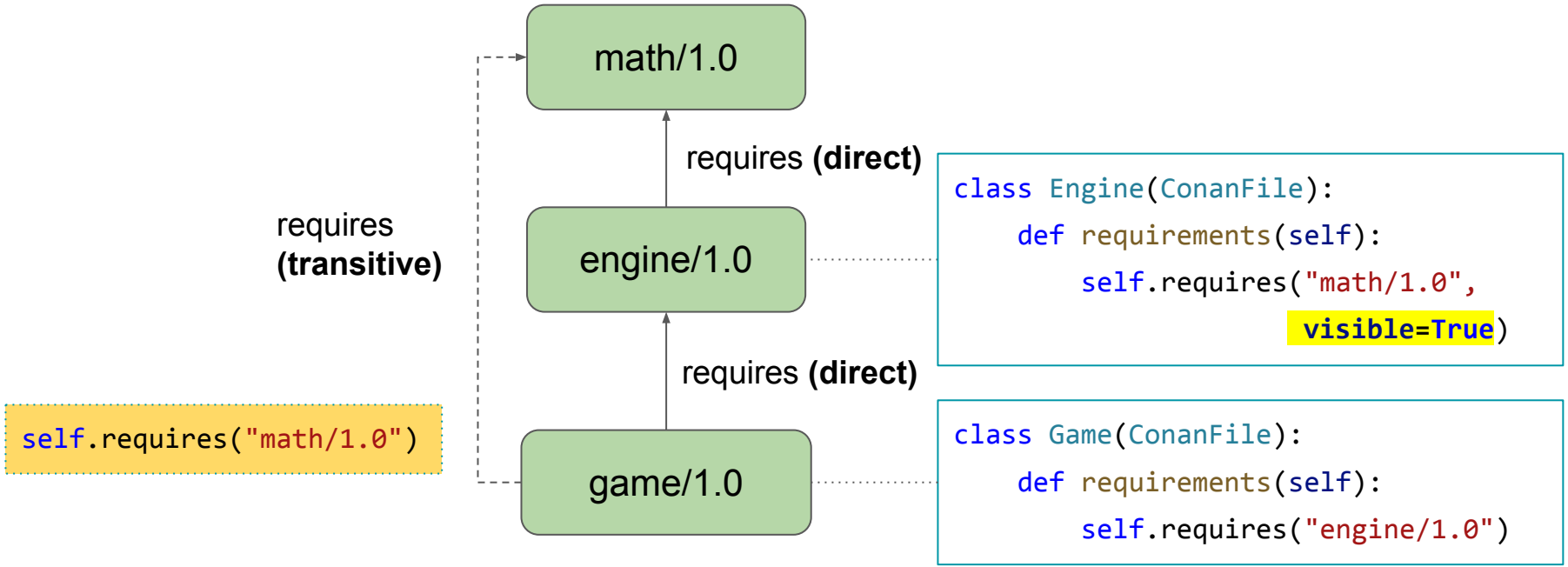
tool_requires = requires

```
def requirements(self):
    self.tool_requires("cmake/1.0")
    # identical to
    self.requires("cmake/1.0",
                  headers=False, libs=False,
                  run=True, build=True,
                  test=False, visible=False,
                  direct=True, package_id_mode=None,
                  ...
                 )
```

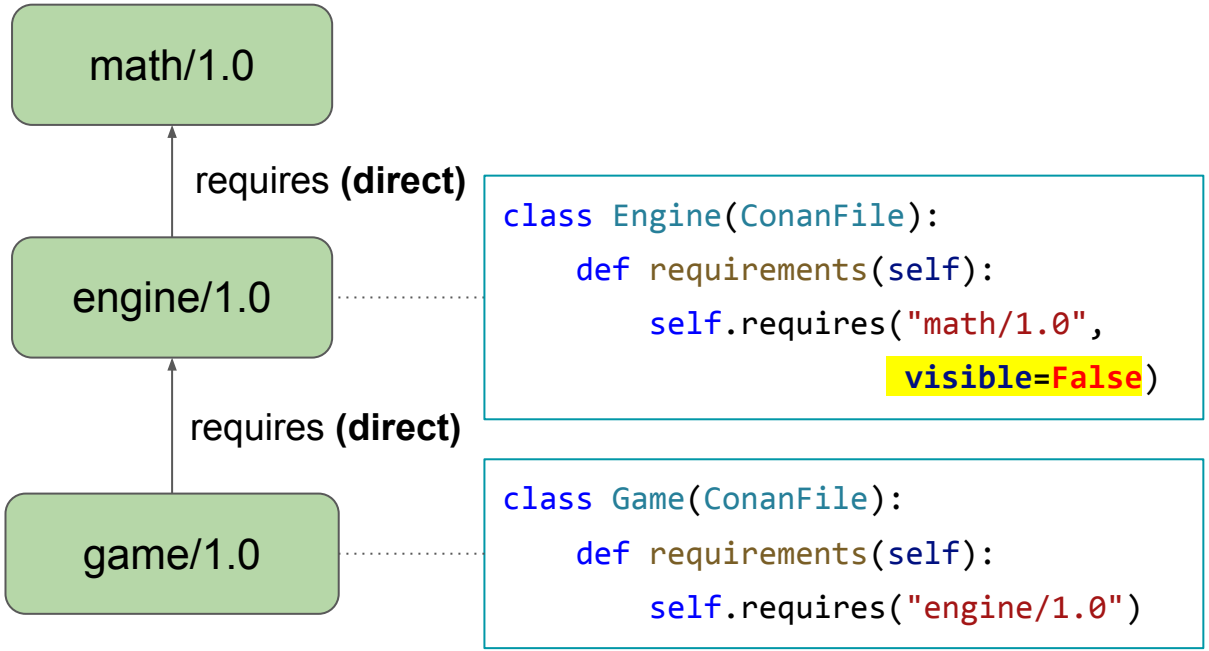
Tool-requires can affect package-id



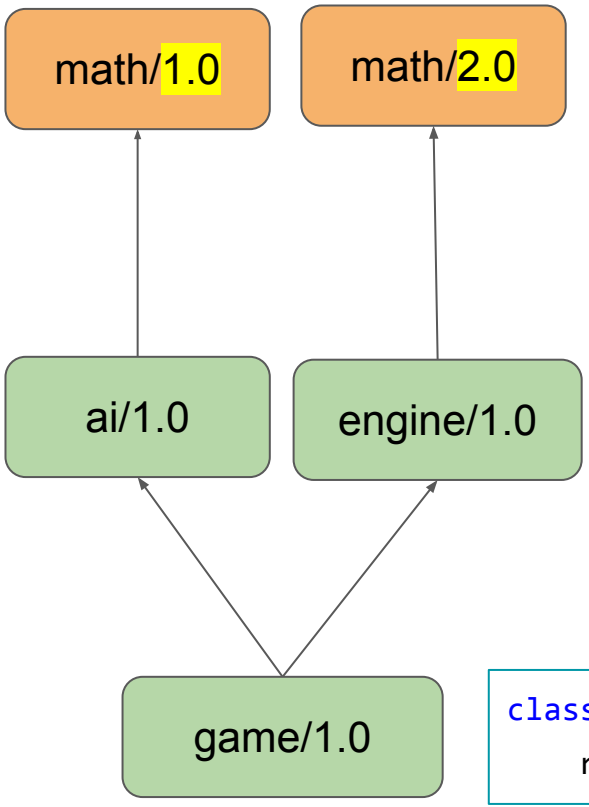
Visible trait



Visible trait



Version conflict



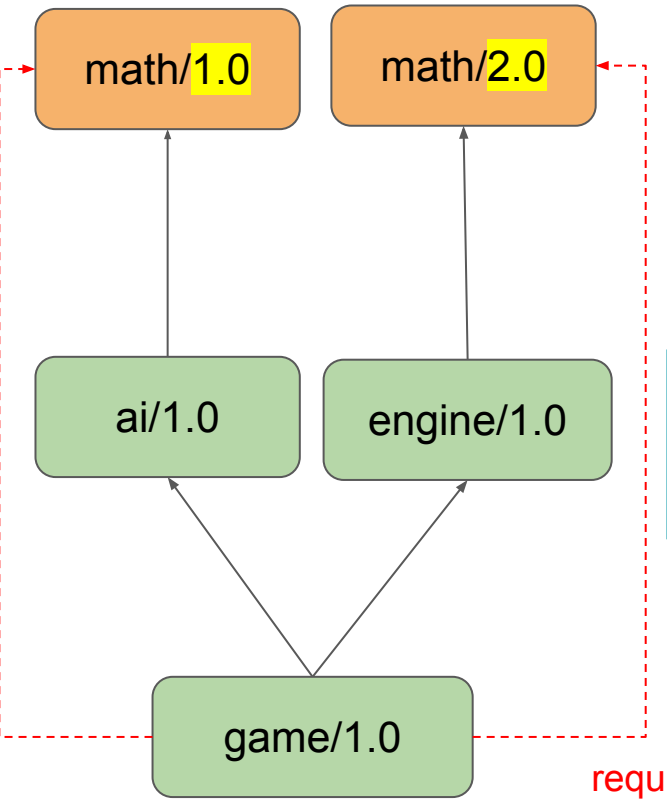
ERROR!

```
class AI(ConanFile):
  def requirements(self):
    self.requires("math/1.0")
```

```
class Engine(ConanFile):
  def requirements(self):
    self.requires("math/2.0")
```

```
class Game(ConanFile):
    requires = "ai/1.0","engine/1.0"
```

Version conflict



ERROR!

```
class AI(ConanFile):
  def requirements(self):
    self.requires("math/1.0")
```

```
class Engine(ConanFile):
  def requirements(self):
    self.requires("math/2.0")
```

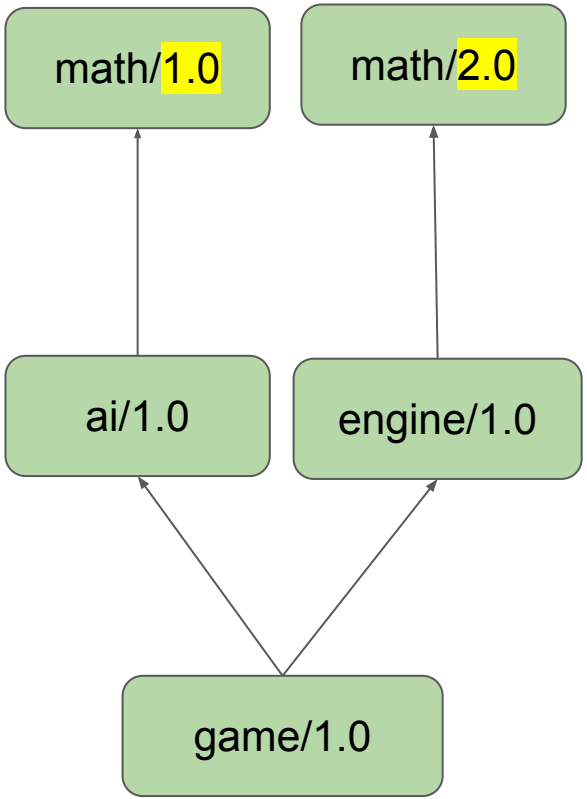
```
self.requires("math/1.0")
```

requires
(transitive)

```
self.requires("math/2.0")
```

requires
(transitive)

Private dependencies: visible trait

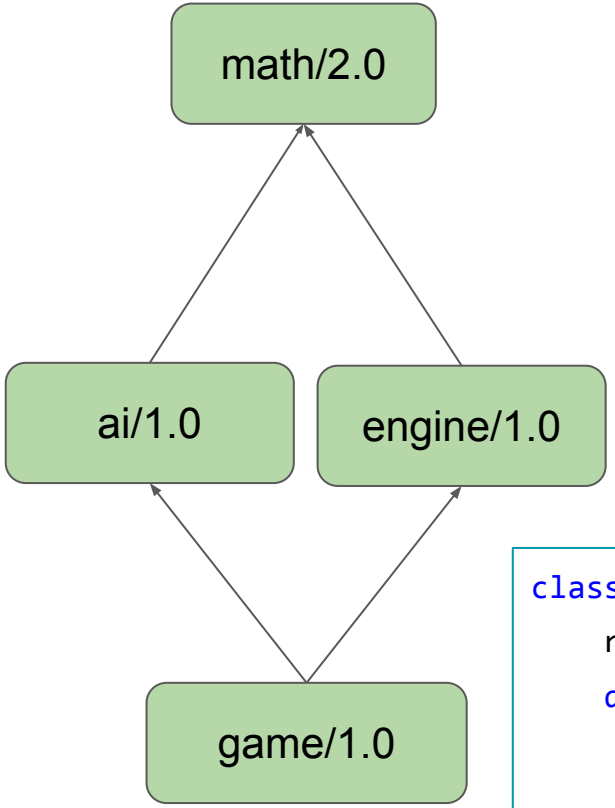


OK!

```
class AI(ConanFile):
  def requirements(self):
    self.requires("math/1.0",
                  visible=False)
```

```
class Engine(ConanFile):
  def requirements(self):
    self.requires("math/2.0",
                  visible=False)
```

Version conflict resolution (override)



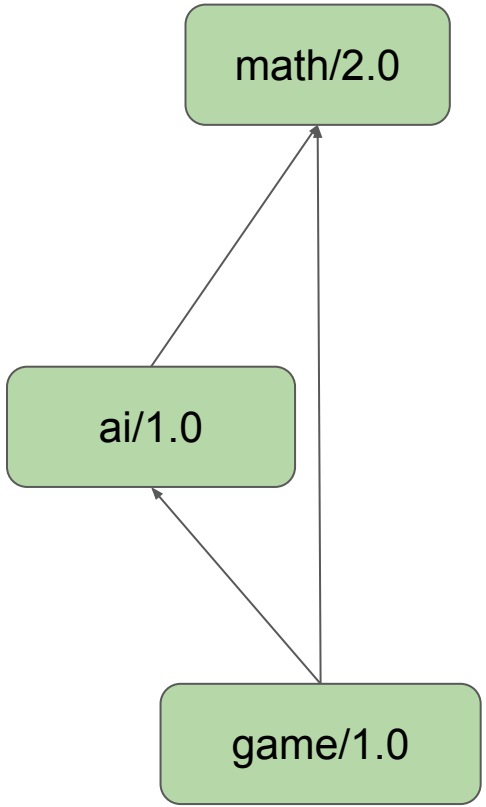

```
class AI(ConanFile):
    requires = "math/1.0"
```

```
class Engine(ConanFile):
    requires = "math/2.0"
```

```
class Game(ConanFile):
    requires = "ai/1.0","engine/1.0"
    def requirements(self):
        self.requires("math/2.0",
                      override=True)
```

Version conflict resolution (force)

```
class AI(ConanFile):
    requires = "math/1.0"
```

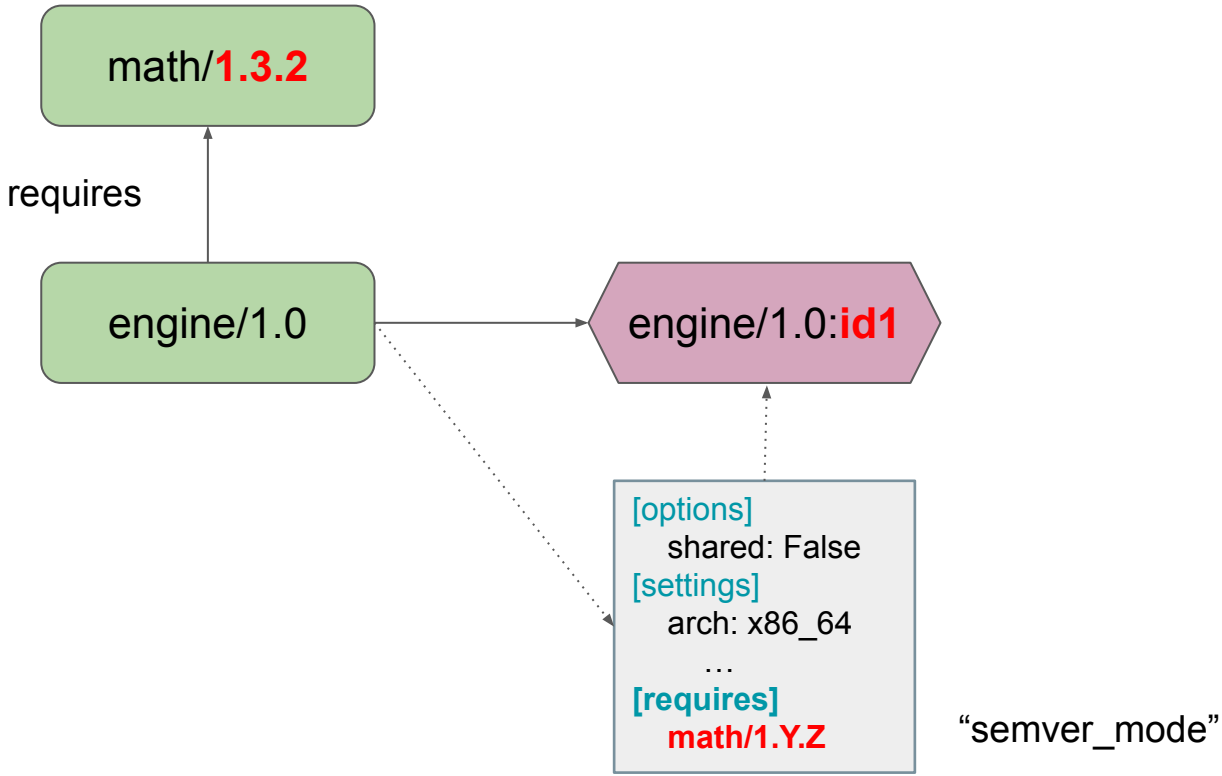



```
class Game(ConanFile):
    requires = "ai/1.0"
    def requirements(self):
        self.requires("math/2.0",
                      force=True)
```

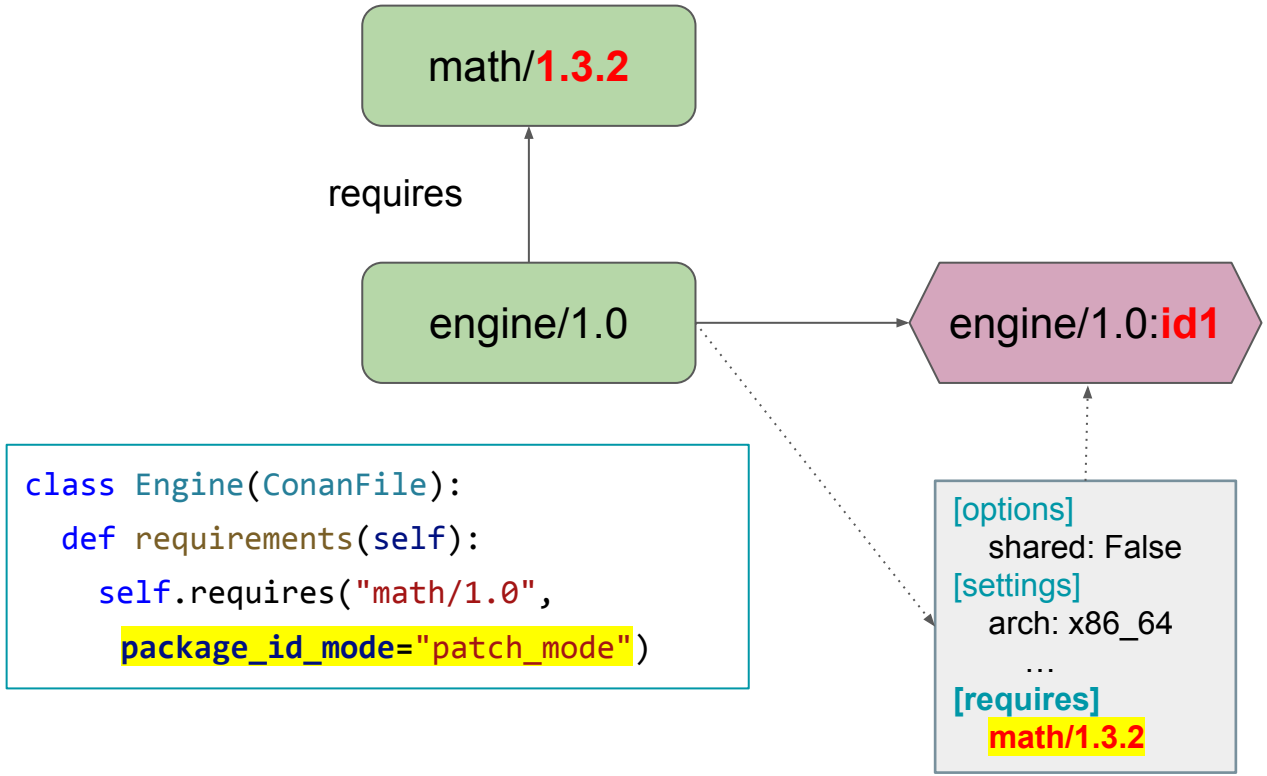
Outline

- The C and C++ native artifacts model
- Conan 1.X packages and dependencies model
- Limitations of Conan 1.X model
- **Conan 2.0 new model**
  - Requirements model
  - Propagation of requirements
  - Headers, libs, run traits
  - Package types
  - Build, visible, override/force traits
  - **New package_id**
- Demo
- Conclusion

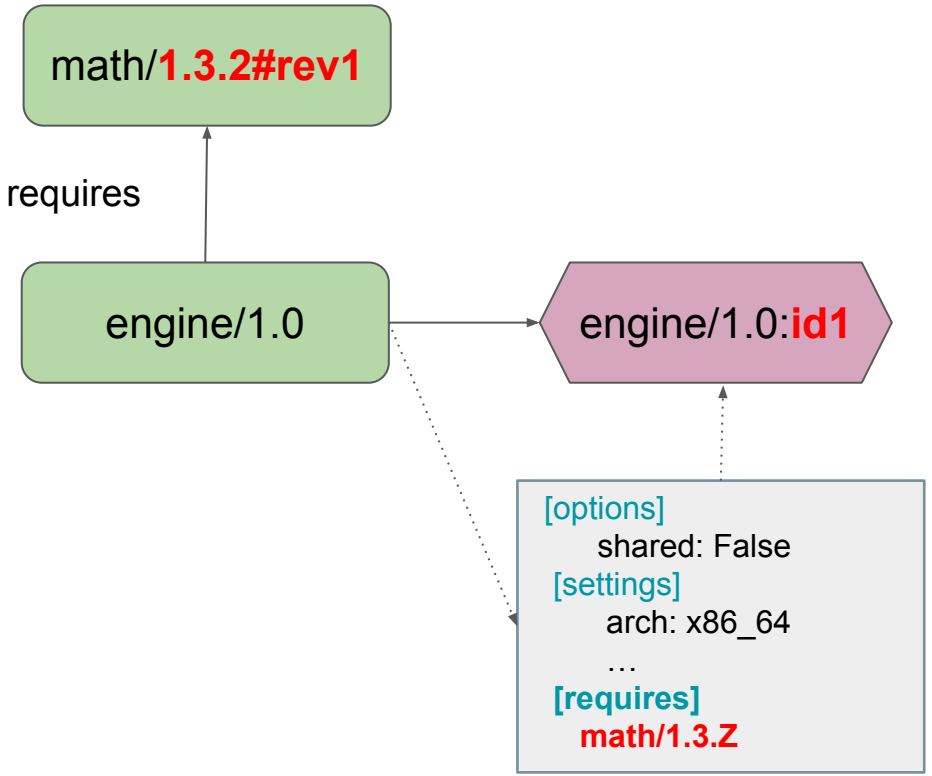
Conan 1.X package_id_mode



Trait package_id_mode



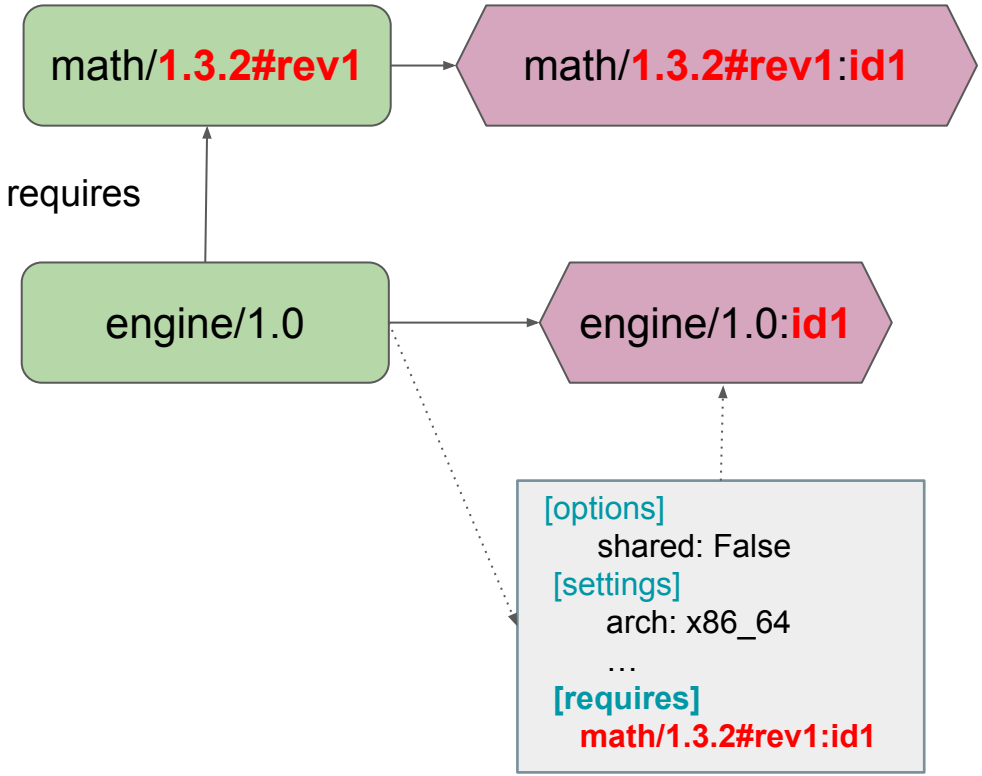
Conan 2.X default package_id modes



non_embed_mode (=minor)

- app→ shared
- shared → shared
- static → static

Conan 2.X default package_id modes



embed_mode (=full)

- app→ static
- app → header
- shared → static
- shared → header
- static → header

Configuring package_id

global.conf

```
core.package_id:default_unknown_mode = semver_mode
core.package_id:default_non_embed_mode = minor_mode
core.package_id:default_embed_mode = full_mode
core.package_id:default_python_mode = minor_mode
core.package_id:default_build_mode = None
```

Outline

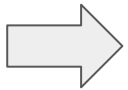
- The C and C++ native artifacts model
- Conan 1.X packages and dependencies model
- Limitations of Conan 1.X model
- Conan 2.0 new model
- **Demo**
- Conclusion

Outline

- The C and C++ native artifacts model
- Conan 1.X packages and dependencies model
- Limitations of Conan 1.X model
- Conan 2.0 new model
- Demo
- **Conclusion**

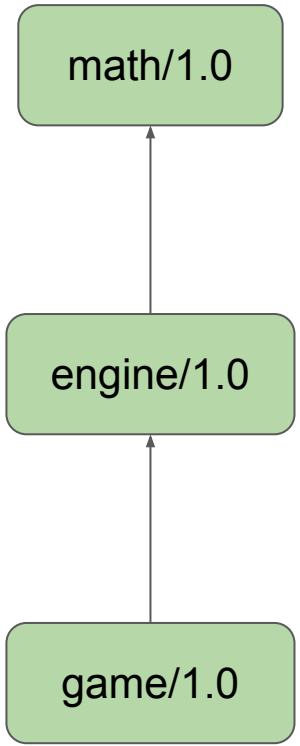
Conclusion

- Require traits:
  - Headers, libs, transitive_headers, transitive_libs, run, build, visible, package_id_mode, options, test
- Package types:
  - static-library, shared-library, header, application, build-script
  - Deducible from “options”
- Requires propagation



- Headers visibility
- Linkage requirements
- “Private” non conflicting dependencies
- Bootstrap cycles
- Much better package_id, knowing when to re-build
- Tool-requires and executables
- Better conflict control and resolution
- Efficient smart skipping unnecessary dependencies

Conclusion



math/conanfile.py

```
class Math(ConanFile):
    name = "math"
    version = "1.0"
    options = {"shared": [True, False]}
```

engine/conanfile.py

```
class Engine(ConanFile):
  options = {"shared": [True, False]}
  def requirements(self):
    self.requires("math/1.0")
```

game/conanfile.py

```
class Game(ConanFile):
  package_type = "application"
  def requirements(self):
    self.requires("engine/1.0")
```

Thank you!

- conan.io (docs, training, github, social)
- Q&A gathertown
- @diegorlosada