the rdyncall package

An improved foreign function interface for R

Daniel Adler
Georg-August
Universität Göttingen

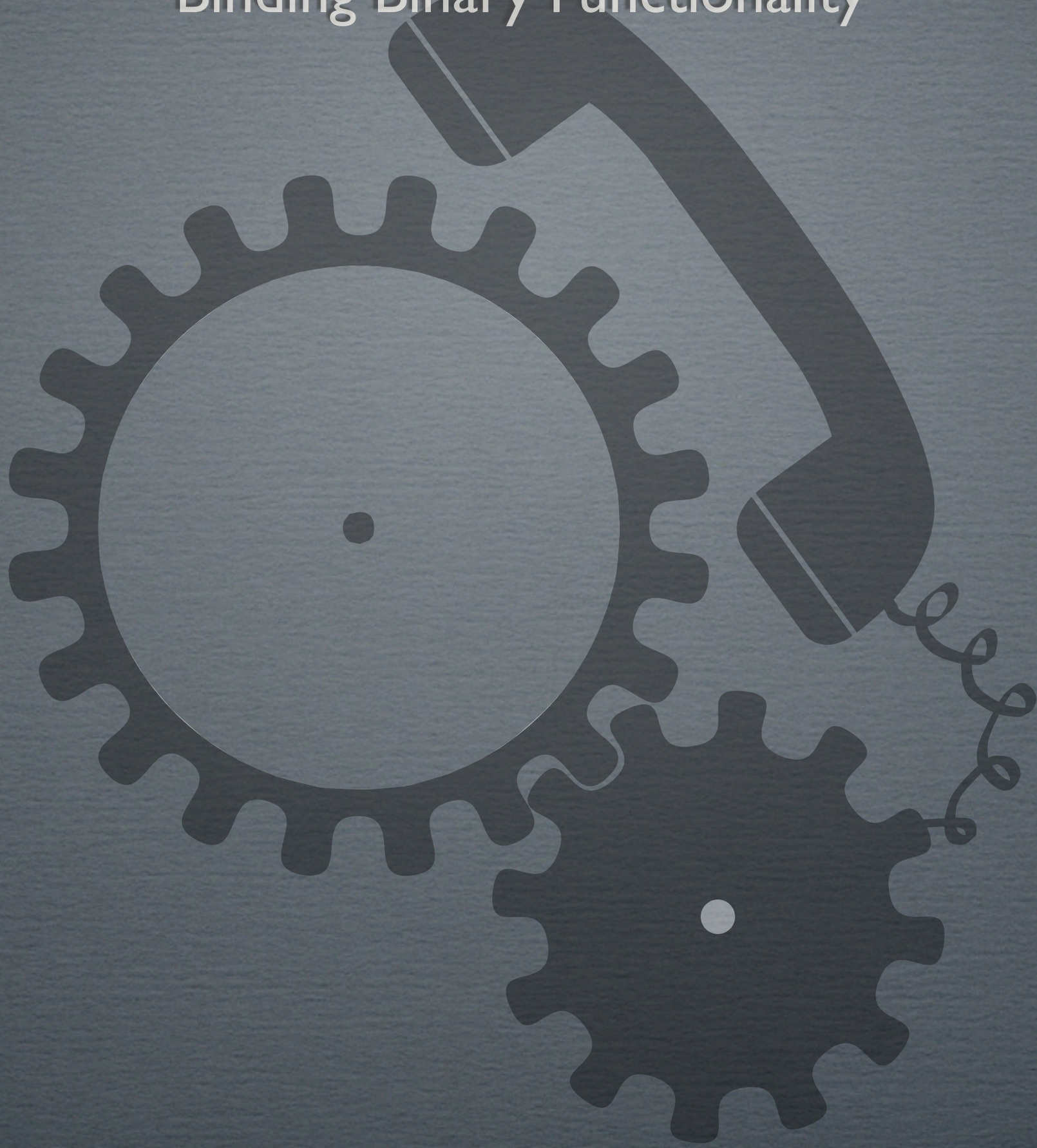
Tassilo Philipp
Potion Studios
Game Development

useR! 2009, 9th of July, Rennes, France

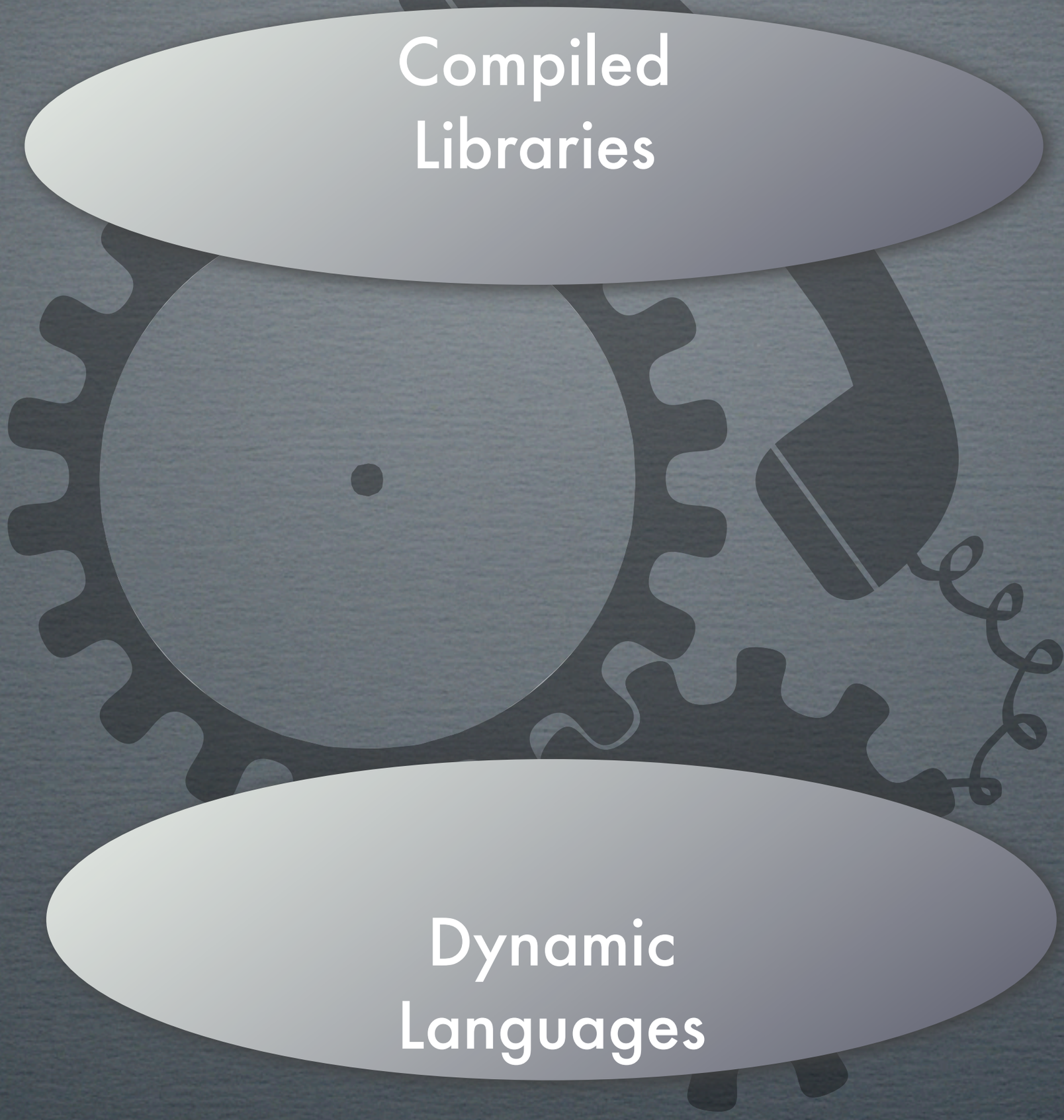
Program of this talk

- Binding Binary Functionality
- Example: Binding R with libSDL using ".C(..)"
- Improvements using ".dyncall(..)"
- Implementation: the dyncall C library
- Overview of the rdyncall R Package
- 'Dynports' concept
- Availability

Binding Binary Functionality

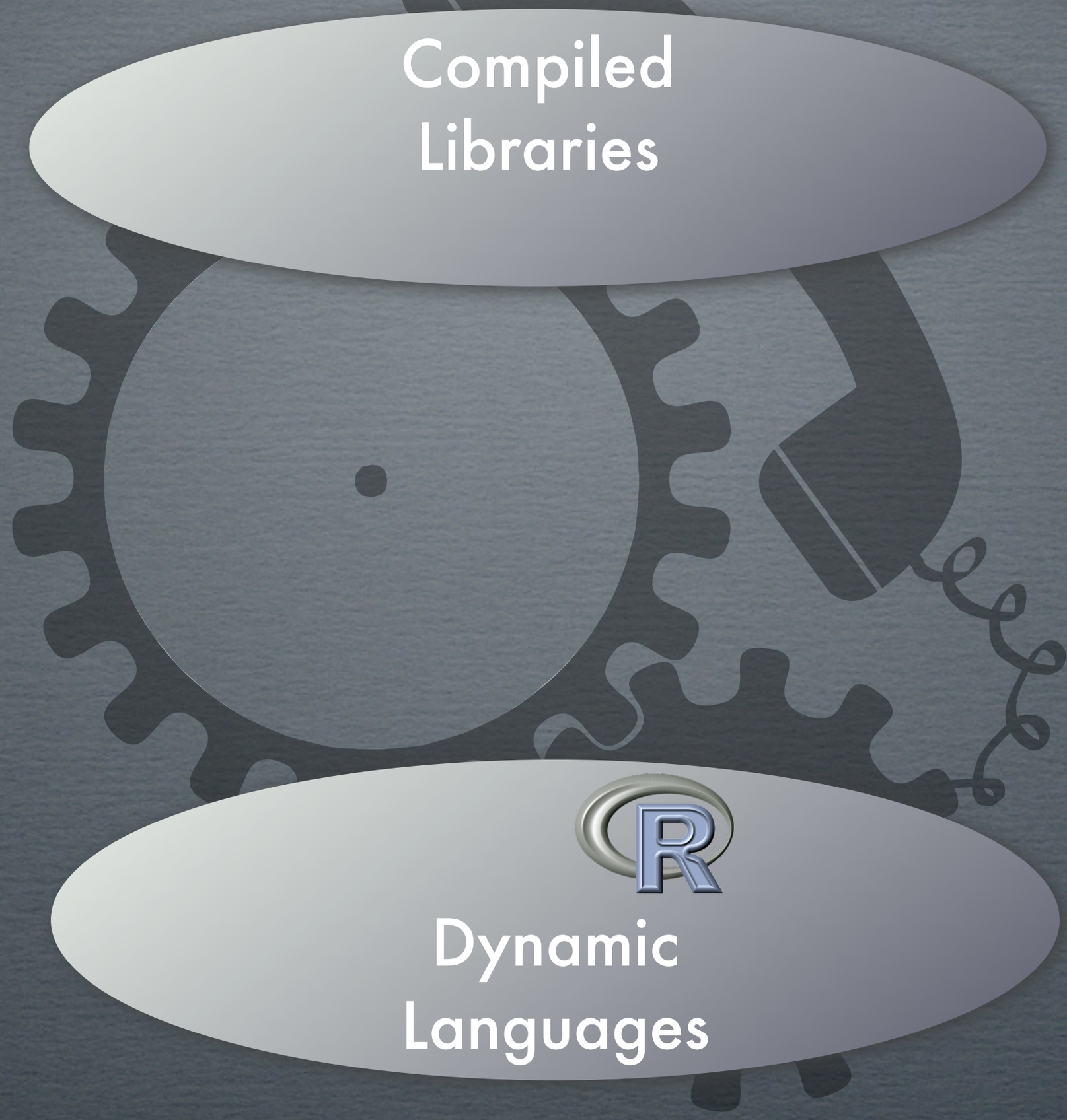


Binding Binary Functionality



Binding Binary Functionality

Compiled
Libraries



Dynamic
Languages

Binding Binary Functionality

Compiled
Libraries

Dynamic
Languages

Binding Binary Functionality

Compiled Libraries

Dynamic Languages

Binding Binary Functionality

Compiled
Libraries

Dynamic
Languages

Binding Binary Functionality

Compiled Libraries

glew C

OpenGL C

SDL Simple Directmedia Layer C

OpenAL C

BULLET PHYSICS LIBRARY C

Dynamic Languages

R

Binding Binary Functionality

Compiled Libraries

glew
C

OpenGL
C

SDL
Simple Directmedia Layer
C

OpenAL
C

BULLET
PHYSICS LIBRARY
C

Dynamic Languages

R

Binding Binary Functionality

Compiled Libraries

glew C

OpenGL C

SDL Simple Directmedia Layer C

OpenAL C

BULLET PHYSICS LIBRARY C

Dynamic Languages

python™

R

Perl

Ruby PROGRAMMING Language

Binding Binary Functionality

Compiled Libraries

glew C

OpenGL C

SDL Simple Directmedia Layer C

OpenAL C

BULLET PHYSICS LIBRARY C

Dynamic Languages

Perl

python™

R

Ruby PROGRAMMING Language

Binding Binary Functionality

Compiled Libraries

glew C

OpenGL C

SDL Simple Directmedia Layer C

openAL C

BULLET PHYSICS LIBRARY C

R C

Dynamic Languages

Perl

python™

R

Ruby PROGRAMMING Language

Binding Binary Functionality

Compiled Libraries

glew — C

OpenGL — C

SDL (Simple Directmedia Layer) — C

OpenAL — C

Bullet Physics Library — C

R — C

Dynamic Languages

Perl

python™

R

Ruby PROGRAMMING Language

Binding Binary Functionality

Compiled Libraries

glew C

OpenGL C

SDL Simple Directmedia Layer C

OpenAL C

BULLET PHYSICS LIBRARY C

R C

Call(args)

Foreign Function Interface

Results

Dynamic Languages

python™

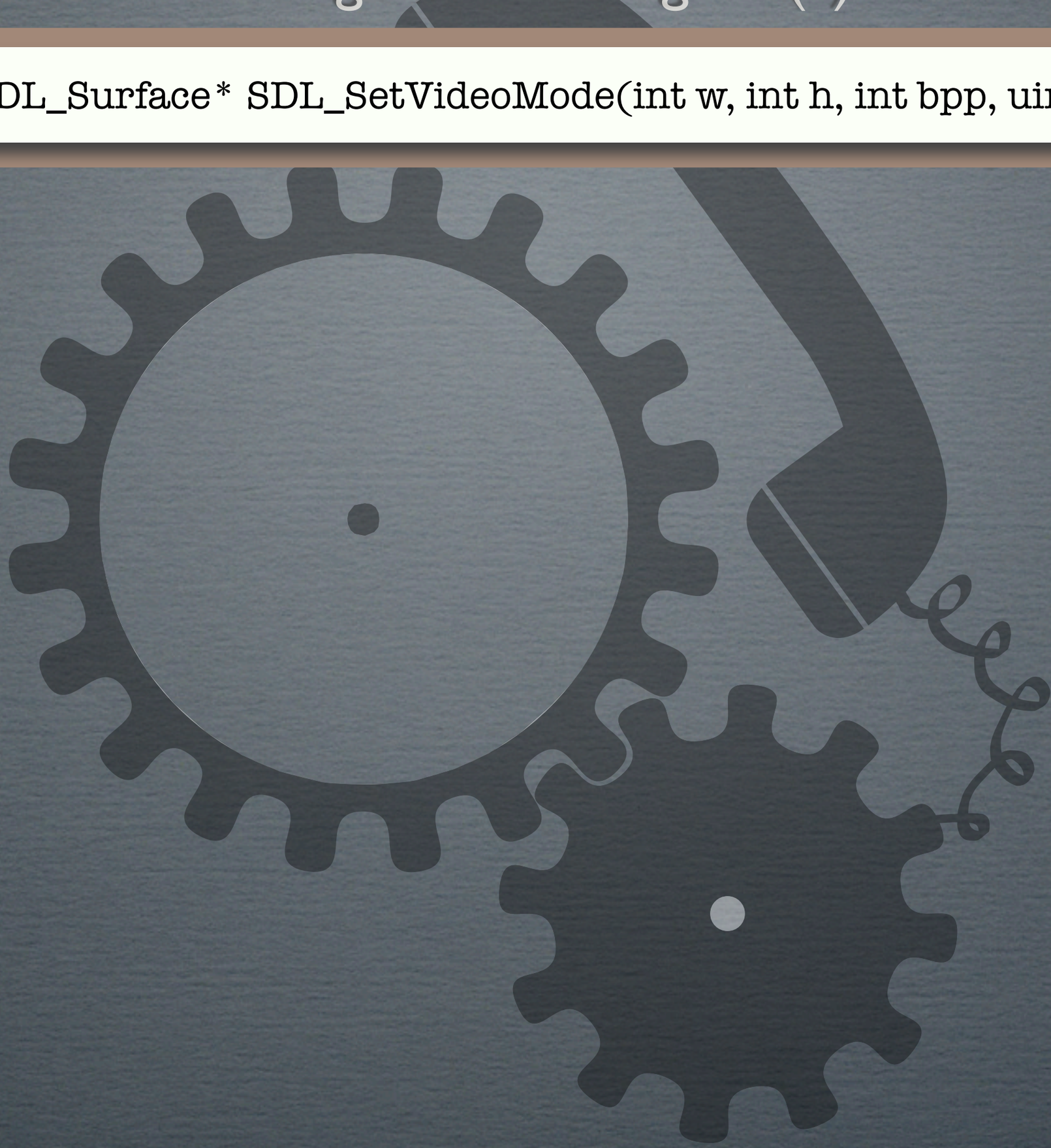
Perl

Ruby PROGRAMMING Language

Binding libSDL using ".C(..)"

C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```



Binding libSDL using ".C(..)"

C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

R calls C:

```
surface <- .C(funaddr, 640L, 480L, 32L, 0L)
```

Binding libSDL using ".C(..)"

C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

R calls C:

```
surface <- .C(funaddr, 640L, 480L, 32L, 0L)
```

fails

Binding libSDL using ".C(..)"

C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

R calls C:

```
surface <- .C(funaddr, 640L, 480L, 32L, 0L)
```

fails

Expected C Interface:

```
void                SDL_SetVideoMode(int* w, int* h, int* bpp, int* flags);
```

Limitations:

R to C mapping:

R Vector mode	C Pointer Type
logical	int*
integer	int*
double	double*
character	char**
raw	unsigned char*

Binding libSDL using ".C(..)"

C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

R calls C:

```
surface <- .C(funaddr, 640L, 480L, 32L, 0L)
```

fails

Expected C Interface:

```
void                SDL_SetVideoMode(int* w, int* h, int* bpp, int* flags);
```

Limitations:

- no scalar types

R to C mapping:

R Vector mode	C Pointer Type
logical	int*
integer	int*
double	double*
character	char**
raw	unsigned char*

Binding libSDL using ".C(..)"

C Function:
```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

R calls C:
```
surface <- .C(funaddr, 640L, 480L, 32L, 0L)
```
fails

Expected C Interface:
```
void                SDL_SetVideoMode(int* w, int* h, int* bpp, int* flags);
```

Limitations:

- no scalar types
- no return types

R to C mapping:

R Vector mode	C Pointer Type
logical	int*
integer	int*
double	double*
character	char**
raw	unsigned char*

Binding libSDL using ".C(..)"

C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

R calls C:

```
surface <- .C(funaddr, 640L, 480L, 32L, 0L)
```

fails

Expected C Interface:

```
void                SDL_SetVideoMode(int* w, int* h, int* bpp, int* flags);
```

Limitations:

- no scalar types
- no return types
- one-to-one mapping

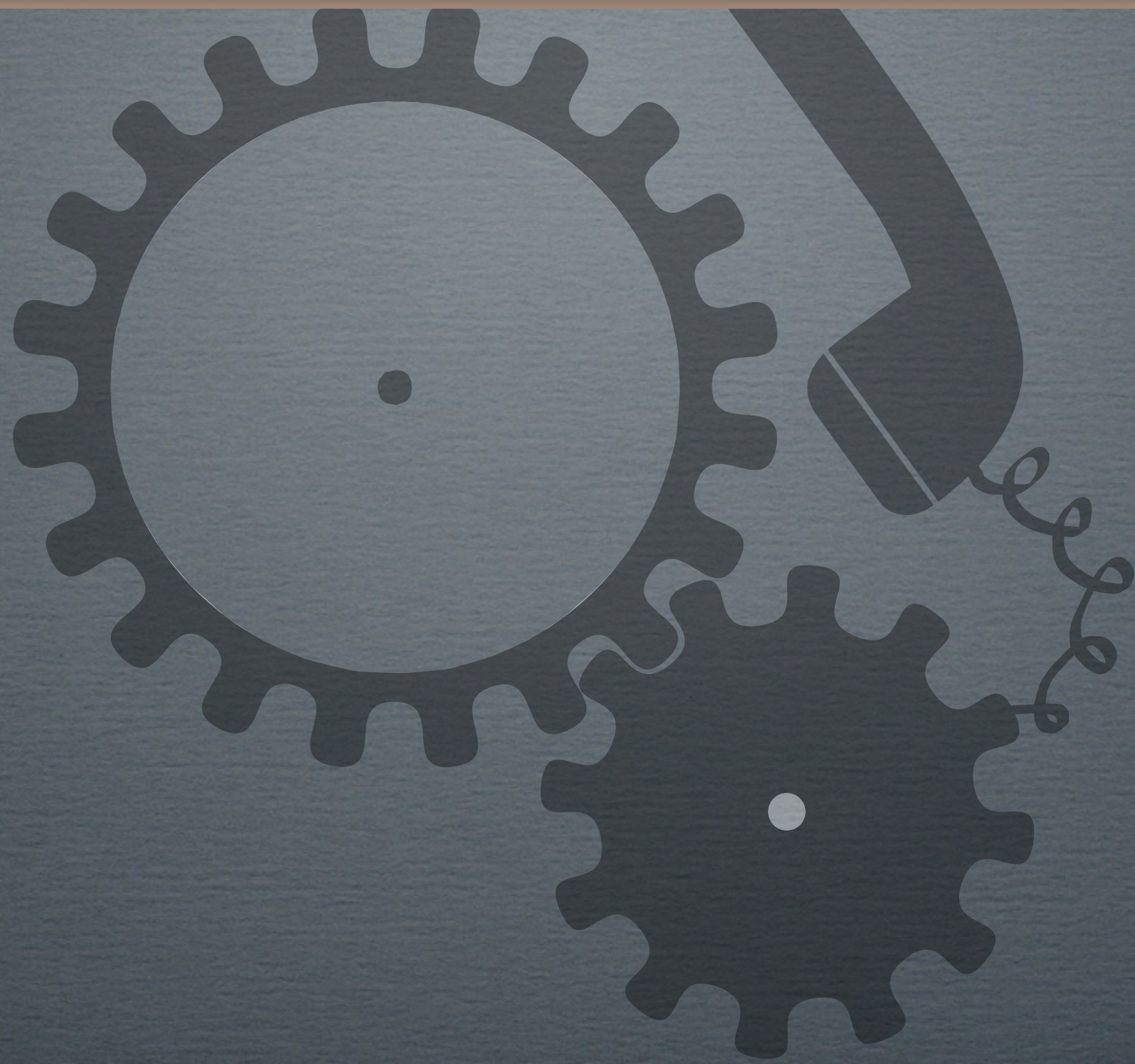
R to C mapping:

R Vector mode	C Pointer Type
logical	int*
integer	int*
double	double*
character	char**
raw	unsigned char*

Binding libSDL using ".Call(..)" + Wrapper DLL

C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```



Binding libSDL using ".Call(..)" + Wrapper DLL

C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

C wrapper source code:

```
SEXP wrapper(SEXP w, SEXP h, SEXP bpp, SEXP flags) {
 return R_MakeExternalPtr(
  SDL_SetVideoMode(
       INTEGER(w)[0], INTEGER(h)[0],
       INTEGER(bpp)[0], (uint) INTEGER(flags)[0]
  ), R_NilValue, R_NilValue);
}
```

Binding libSDL using ".Call(..)" + Wrapper DLL

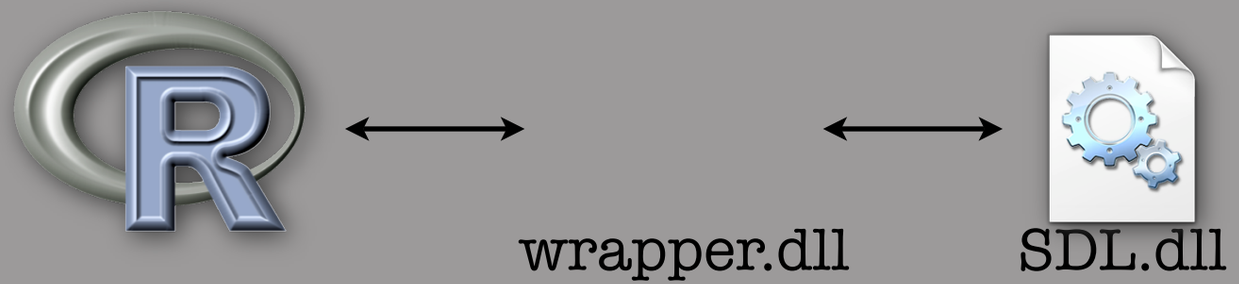
C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

C wrapper source code:

```
SEXP wrapper(SEXP w, SEXP h, SEXP bpp, SEXP flags) {
 return R_MakeExternalPtr(
  SDL_SetVideoMode(
       INTEGER(w)[0], INTEGER(h)[0],
       INTEGER(bpp)[0], (uint) INTEGER(flags)[0]
  ), R_NilValue, R_NilValue);
}
```

Runtime:



Binding libSDL using ".Call(..)" + Wrapper DLL

C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

C wrapper source code:

```
SEXP wrapper(SEXP w, SEXP h, SEXP bpp, SEXP flags) {
 return R_MakeExternalPtr(
  SDL_SetVideoMode(
       INTEGER(w)[0], INTEGER(h)[0],
       INTEGER(bpp)[0], (uint) INTEGER(flags)[0]
  ), R_NilValue, R_NilValue);
}
```

compile, link

Runtime:

wrapper.dll

SDL.dll

Binding libSDL using ".Call(..)" + Wrapper DLL

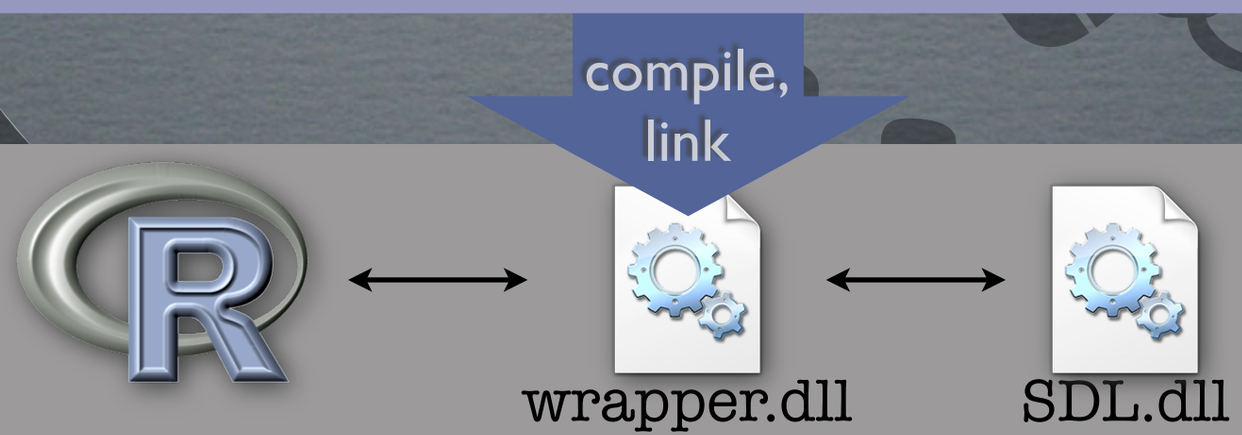
C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

C wrapper source code:

```
SEXP wrapper(SEXP w, SEXP h, SEXP bpp, SEXP flags) {
 return R_MakeExternalPtr(
  SDL_SetVideoMode(
       INTEGER(w)[0], INTEGER(h)[0],
       INTEGER(bpp)[0], (uint) INTEGER(flags)[0]
  ), R_NilValue, R_NilValue);
}
```

Runtime:



Binding libSDL using ".Call(..)" + Wrapper DLL

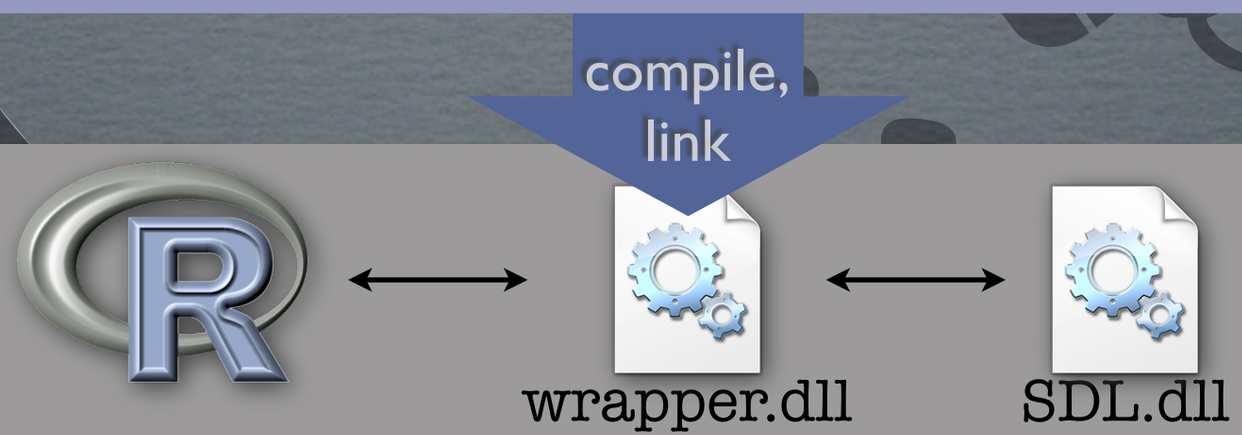
C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

C wrapper source code:

```
SEXP wrapper(SEXP w, SEXP h, SEXP bpp, SEXP flags) {
 return R_MakeExternalPtr(
  SDL_SetVideoMode(
       INTEGER(w)[0], INTEGER(h)[0],
       INTEGER(bpp)[0], (uint) INTEGER(flags)[0]
  ), R_NilValue, R_NilValue);
}
```

Runtime:



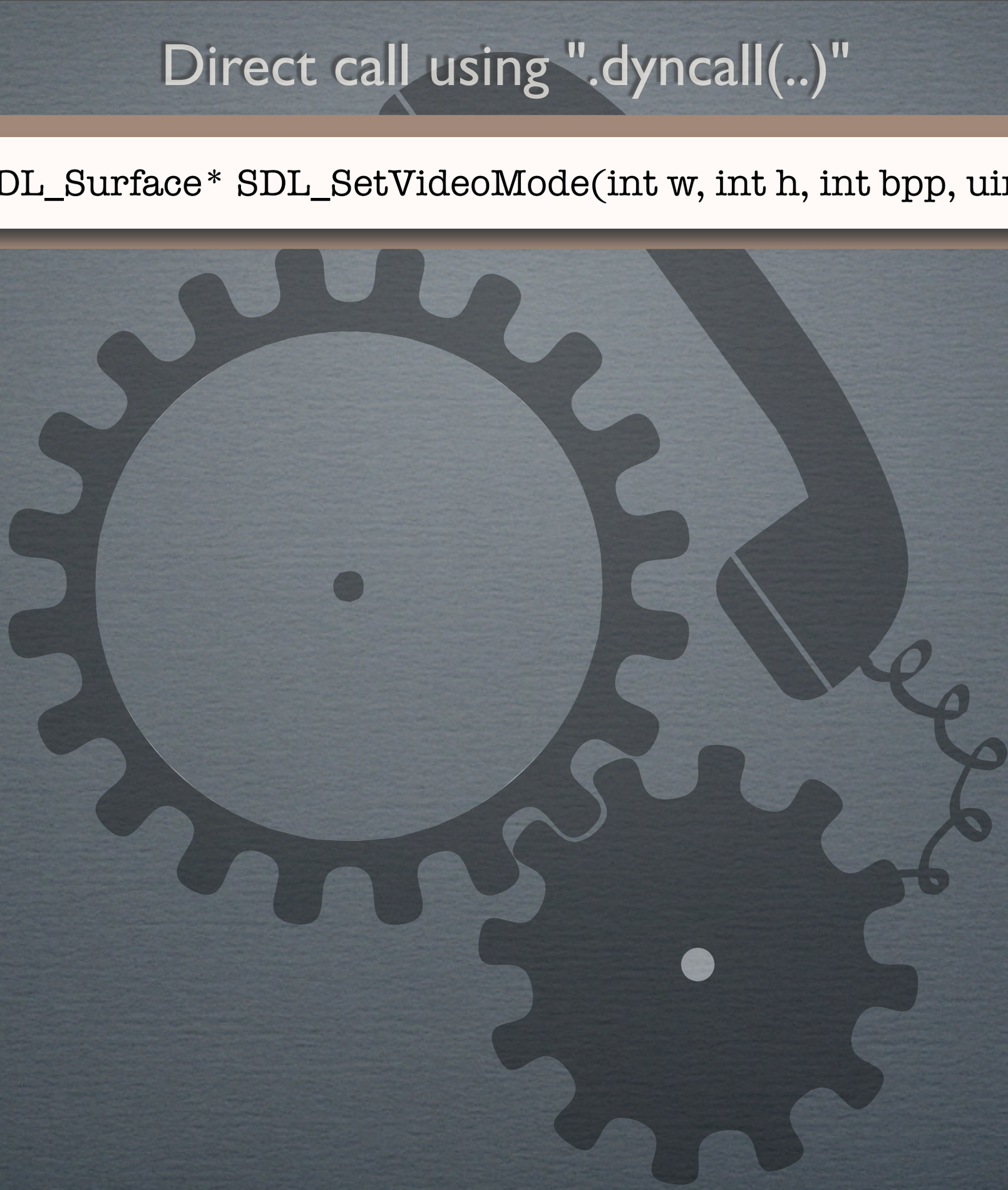
R calls C:

```
surfPtr <- .Call(wrapper_funptr, 640L, 480L, 32L, 0L )
```

Direct call using ".dyncall(..)"

C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```



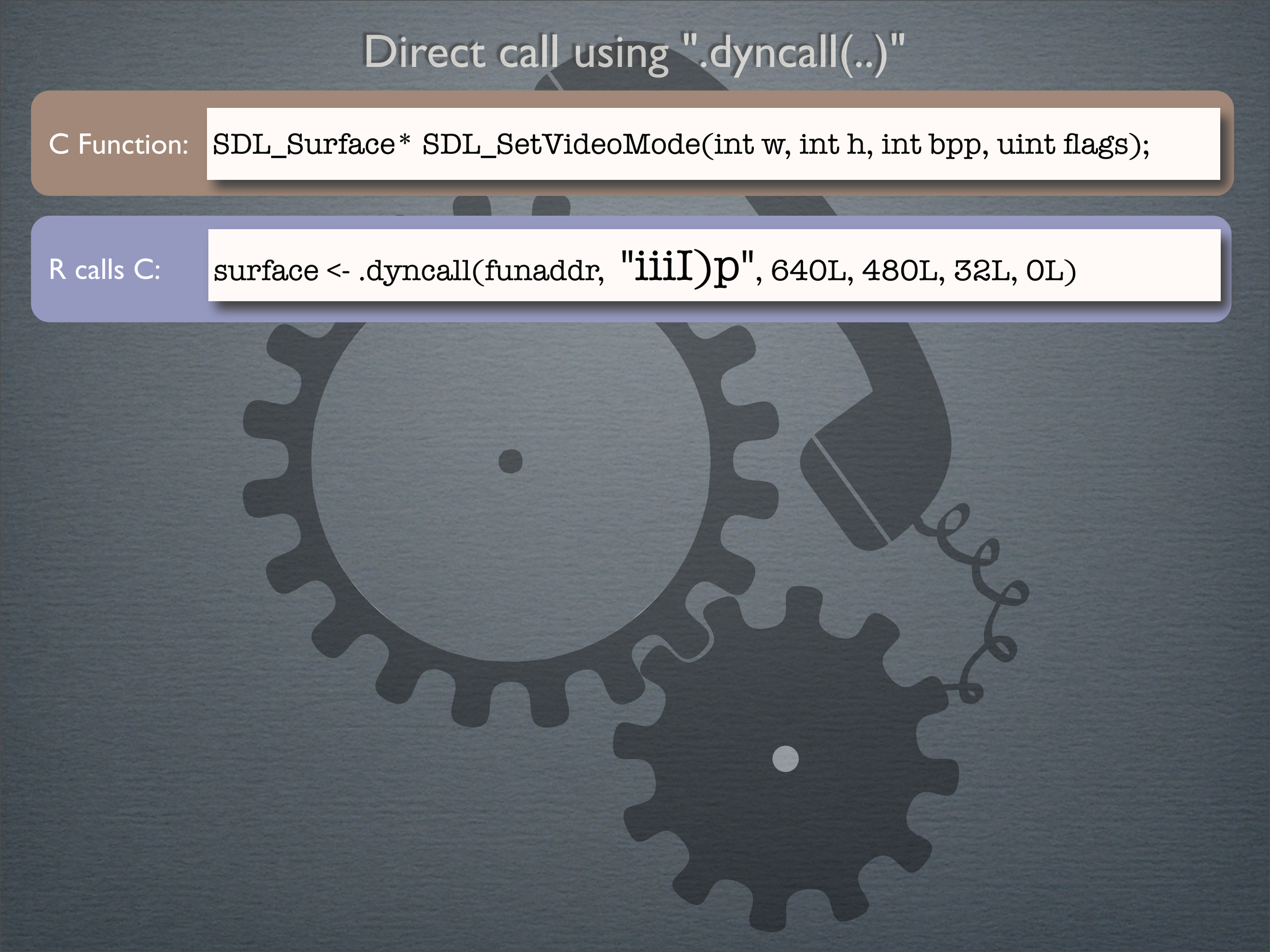
Direct call using ".dyncall(..)"

C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

R calls C:

```
surface <- .dyncall(funaddr, "iiiI)p", 640L, 480L, 32L, 0L)
```



Direct call using ".dyncall(..)"

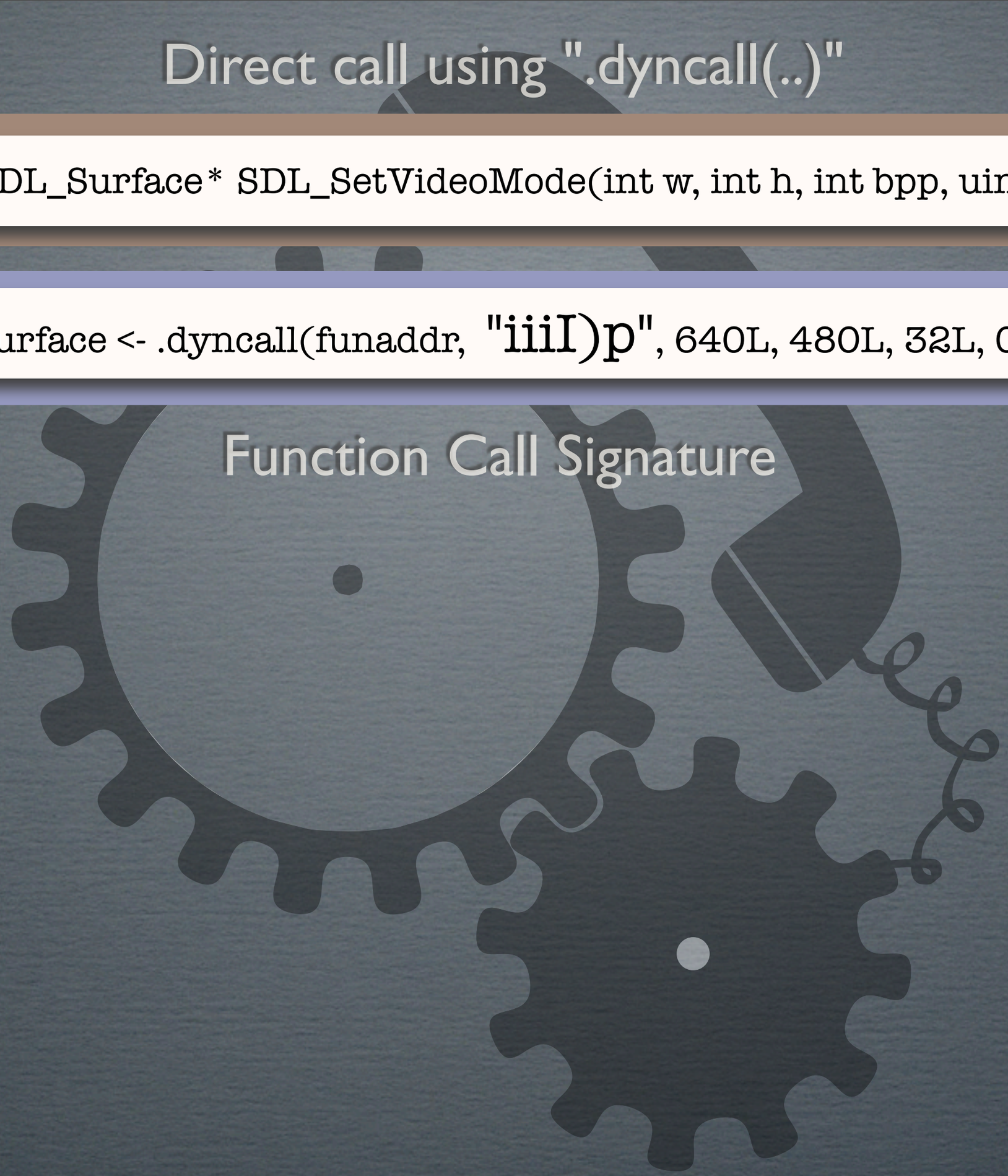
C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

R calls C:

```
surface <- .dyncall(funaddr, "iiiI)p", 640L, 480L, 32L, 0L)
```

Function Call Signature



Direct call using ".dyncall(..)"

C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

R calls C:

```
surface <- .dyncall(funaddr, "iiiI)p", 640L, 480L, 32L, 0L)
```

Function Call Signature

Signature Format:

"{argument types in left-to-right order} ')' {return type}"

Direct call using ".dyncall(..)"

C Function: `SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);`

R calls C: `surface <- .dyncall(funaddr, "iiiI)p", 640L, 480L, 32L, 0L)`

Function Call Signature

Signature Format: "{argument types in left-to-right order} ')' {return type}"

C Type	Signature Character
void	'v'
char, short, int, long, long long	'c', 's', 'i', 'j', 'l'
unsigned *integers (capitalized)*	'C', 'S', 'I', 'J', 'L'
float, double	'f', 'd'
T (any C pointer)*	'p' or '*' ...
const char* (C String)	'Z'
bool (c++), _Bool_t	'B'
struct/union *C pointers*	'*<' *typename* '>'

Direct call using ".dyncall(..)"

C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

R calls C:

```
surface <- .dyncall(funaddr, "iiiI)p", 640L, 480L, 32L, 0L)
```

Function Call Signature

Signature Format:

"{argument types in left-to-right order} ')' {return type}"

Supports

C Type	Signature Character
void	'v'
char, short, int, long, long long	'c', 's', 'i', 'j', 'l'
unsigned *integers (capitalized)*	'C', 'S', 'I', 'J', 'L'
float, double	'f', 'd'
T (any C pointer)*	'p' or '*' ...
const char* (C String)	'Z'
bool (c++), _Bool_t	'B'
struct/union *C pointers*	'*<' *typename* '>'

Direct call using ".dyncall(..)"

C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

R calls C:

```
surface <- .dyncall(funaddr, "iiiI)p", 640L, 480L, 32L, 0L)
```

Function Call Signature

Signature Format:

"{argument types in left-to-right order} ')' {return type}"

Supports

- scalars

C Type	Signature Character
void	'v'
char, short, int, long, long long	'c', 's', 'i', 'j', 'l'
unsigned *integers (capitalized)*	'C', 'S', 'I', 'J', 'L'
float, double	'f', 'd'
T (any C pointer)*	'p' or '*' ...
const char* (C String)	'Z'
bool (c++), _Bool_t	'B'
struct/union *C pointers*	'*<' *typename* '>'

Direct call using ".dyncall(..)"

C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

R calls C:

```
surface <- .dyncall(funaddr, "iiiI)p", 640L, 480L, 32L, 0L)
```

Function Call Signature

Signature Format:

"{argument types in left-to-right order} ')' {return type}"

Supports

- scalars
- return types

C Type	Signature Character
void	'v'
char, short, int, long, long long	'c', 's', 'i', 'j', 'l'
unsigned *integers (capitalized)*	'C', 'S', 'I', 'J', 'L'
float, double	'f', 'd'
T (any C pointer)*	'p' or '*' ...
const char* (C String)	'Z'
bool (c++), _Bool_t	'B'
struct/union *C pointers*	'*<' *typename* '>'

Direct call using ".dyncall(..)"

C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

R calls C:

```
surface <- .dyncall(funaddr, "iiiI)p", 640L, 480L, 32L, 0L)
```

Function Call Signature

Signature Format:

"{argument types in left-to-right order} ')' {return type}"

Supports

- scalars
- return types
- atomic data ptrs

C Type	Signature Character
void	'v'
char, short, int, long, long long	'c', 's', 'i', 'j', 'l'
unsigned *integers (capitalized)*	'C', 'S', 'I', 'J', 'L'
float, double	'f', 'd'
T (any C pointer)*	'p' or '*' ...
const char* (C String)	'Z'
bool (c++), _Bool_t	'B'
struct/union *C pointers*	'*<' *typename* '>'

Direct call using ".dyncall(..)"

C Function:

```
SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);
```

R calls C:

```
surface <- .dyncall(funaddr, "iiiI)p", 640L, 480L, 32L, 0L)
```

Function Call Signature

Signature Format:

"{argument types in left-to-right order} ')' {return type}"

Supports

- scalars
- return types
- atomic data ptrs
- 8..64 bit (u)ints

C Type	Signature Character
void	'v'
char, short, int, long, long long	'c', 's', 'i', 'j', 'l'
unsigned *integers (capitalized)*	'C', 'S', 'I', 'J', 'L'
float, double	'f', 'd'
T (any C pointer)*	'p' or '*' ...
const char* (C String)	'Z'
bool (c++), _Bool_t	'B'
struct/union *C pointers*	'*<' *typename* '>'

Direct call using ".dyncall(..)"

C Function: `SDL_Surface* SDL_SetVideoMode(int w, int h, int bpp, uint flags);`

R calls C: `surface <- .dyncall(funaddr, "iiiI)p", 640L, 480L, 32L, 0L)`

Function Call Signature

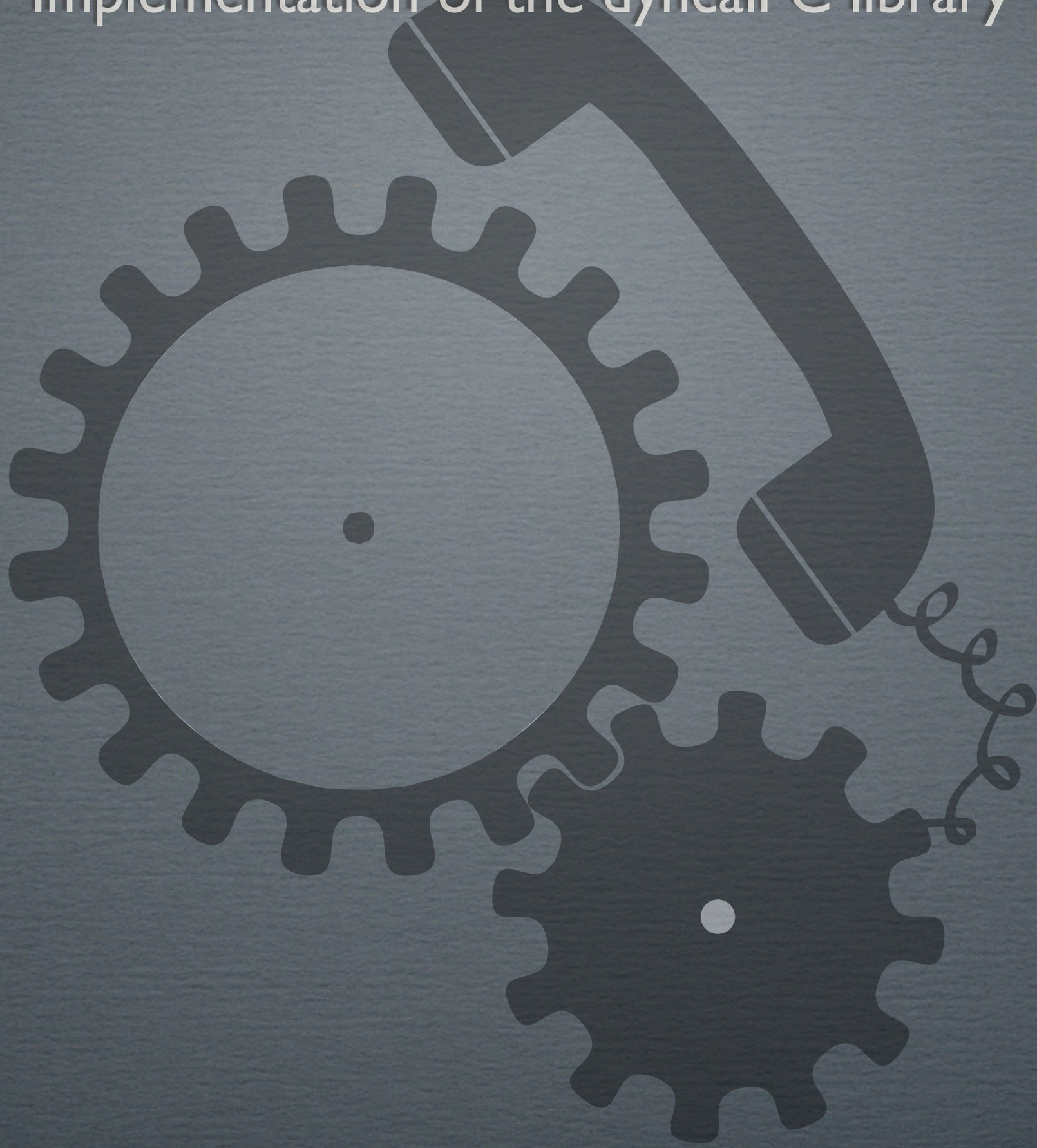
Signature Format: "{argument types in left-to-right order} ')' {return type}"

Supports

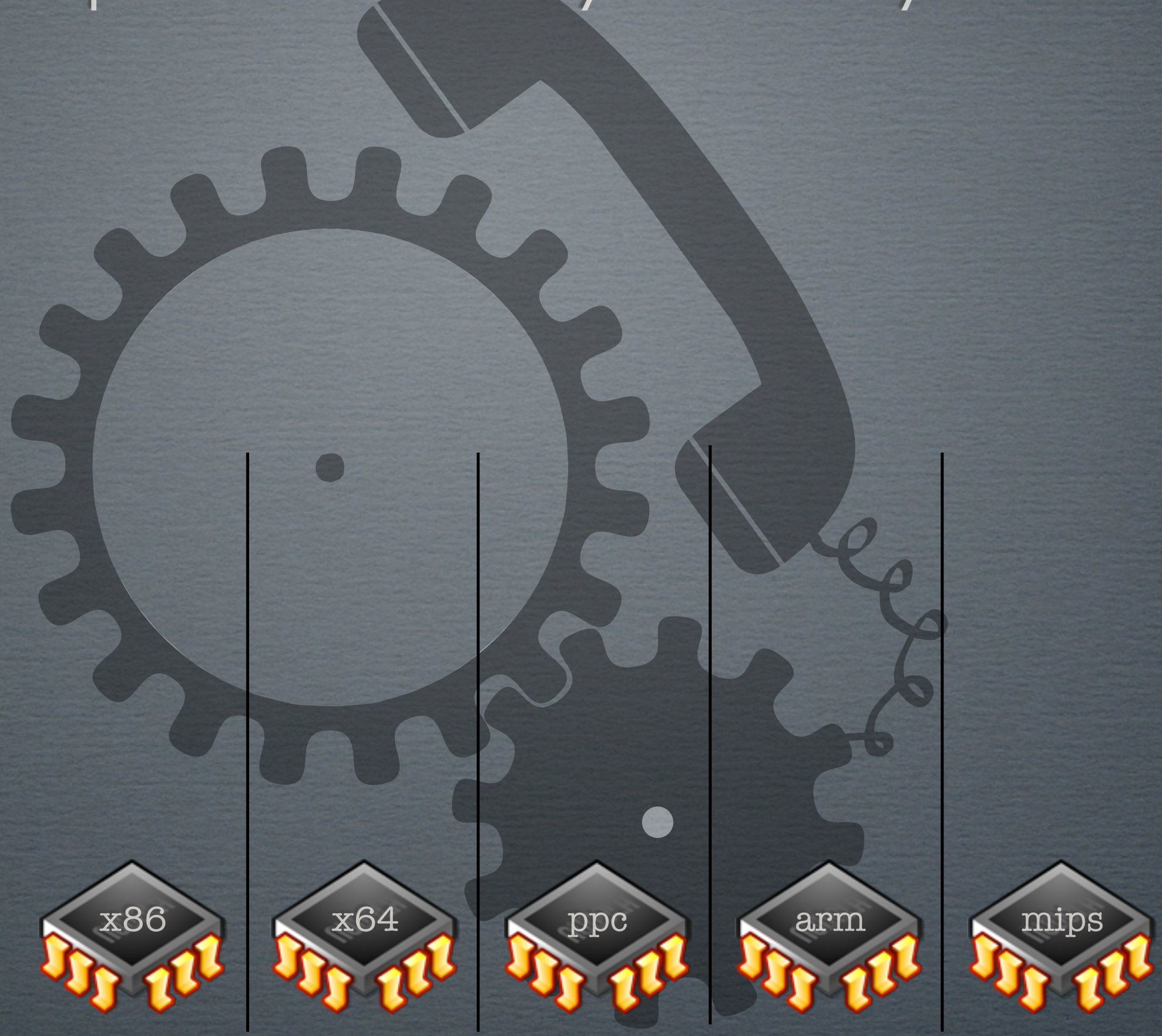
- scalars
- return types
- atomic data ptrs
- 8..64 bit (u)ints
- struct/union ptrs

C Type	Signature Character
void	'v'
char, short, int, long, long long	'c', 's', 'i', 'j', 'l'
unsigned *integers (capitalized)*	'C', 'S', 'I', 'J', 'L'
float, double	'f', 'd'
T (any C pointer)*	'p' or '*' ...
const char* (C String)	'Z'
bool (c++), _Bool_t	'B'
struct/union *C pointers*	'*<' *typename* '>'

Implementation of the dyncall C library



Implementation of the dyncall C library



Implementation of the dyncall C library

Platforms

Processors

Implementation of the dyncall C library

C/C++ Compilers

Platforms

Processors

x86 x64 ppc arm mips

Implementation of the dyncall C library

Calling Conventions	cdecl stdcall fastcall.gcc fastcall.ms thiscall.gcc thiscall.ms	win64 system v	darwin system v	arm9e thumb	eabi
C/C++ Compilers			GCC		
Platforms			NetBSD SOLARIS		
Processors	x86	x64	ppc	arm	mips

Implementation of the dyncall C library

dyncall library

Calling Conventions

cdecl
stdcall
fastcall.gcc
fastcall.ms
thiscall.gcc
thiscall.ms

win64
system v

darwin
system v

arm9e
thumb

eabi

C/C++ Compilers

Platforms

Processors

Implementation of the dyncall C library

C Interface

dyncall library

Calling Conventions

cdecl stdcall fastcall.gcc fastcall.ms thiscall.gcc thiscall.ms	win64 system v	darwin system v	arm9e thumb	eabi

C/C++ Compilers


Platforms

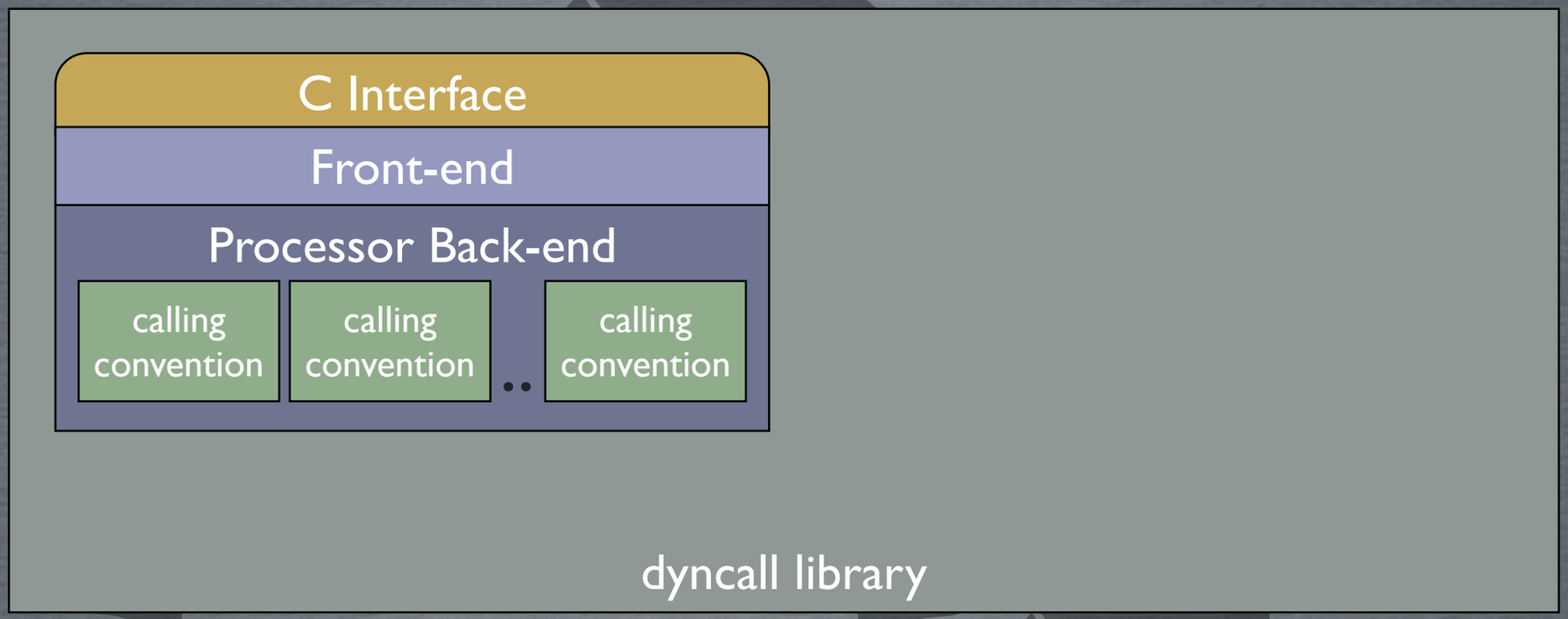






Processors

Implementation of the dyncall C library



Calling Conventions

cdecl
stdcall
fastcall.gcc
fastcall.ms
thiscall.gcc
thiscall.ms

win64
system v

darwin
system v

arm9e
thumb

eabi

C/C++ Compilers

GCC

Platforms

NetBSD

SOLARIS

Processors

x86

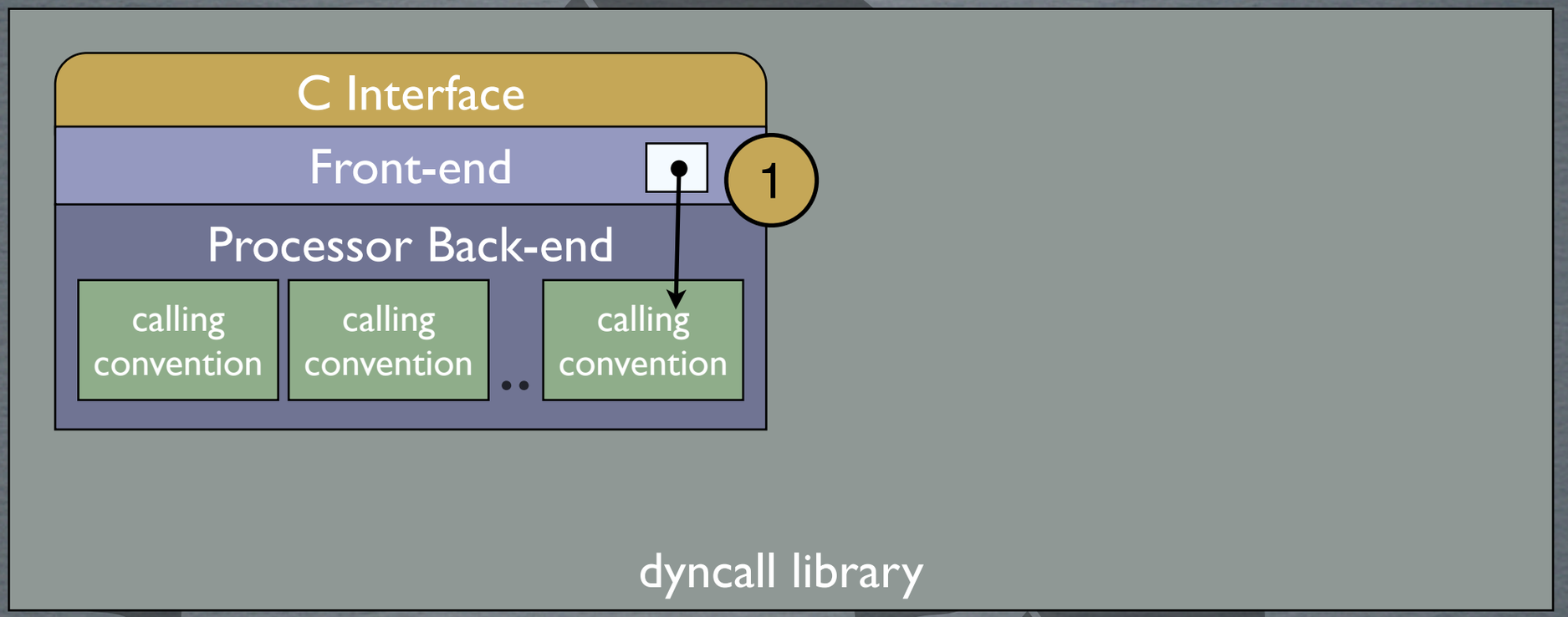
x64

ppc

arm

mips

Implementation of the dyncall C library



Calling Conventions

cdecl
stdcall
fastcall.gcc
fastcall.ms
thiscall.gcc
thiscall.ms

win64
system v

darwin
system v

arm9e
thumb

eabi

C/C++ Compilers

Platforms

Processors

x86

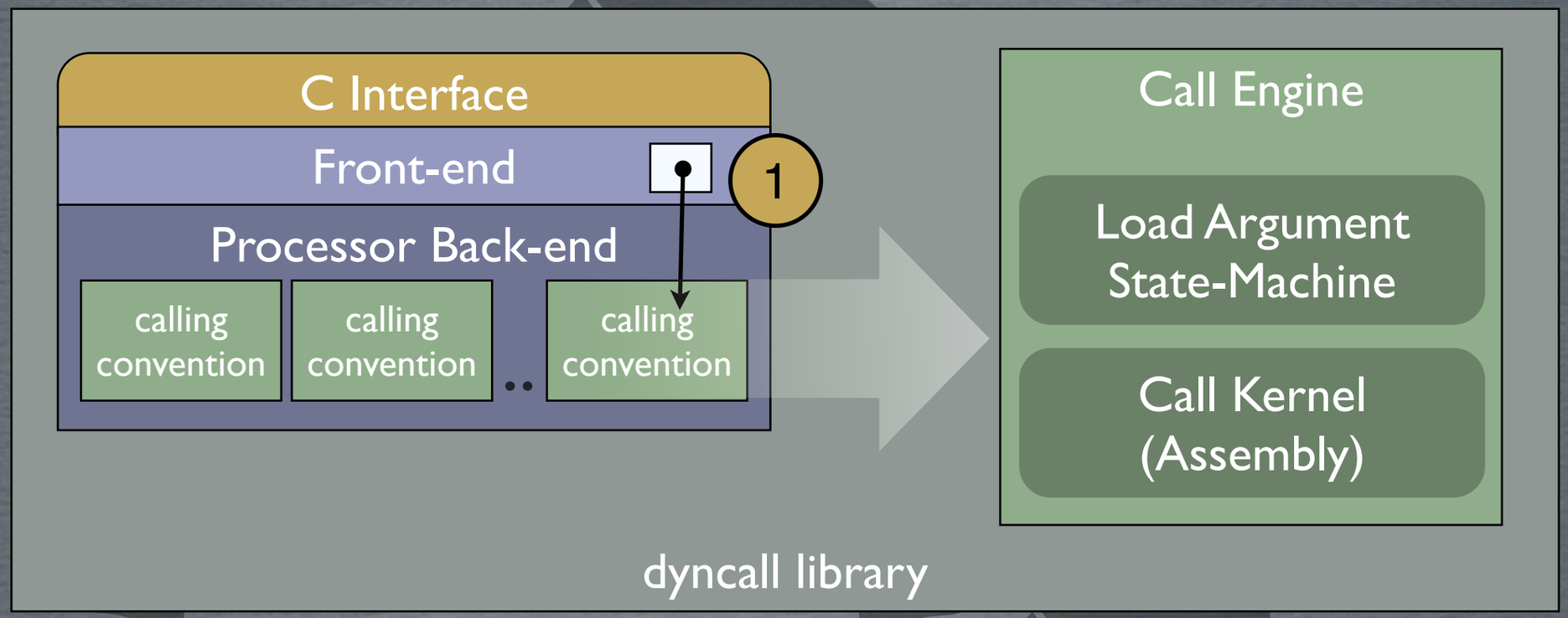
x64

ppc

arm

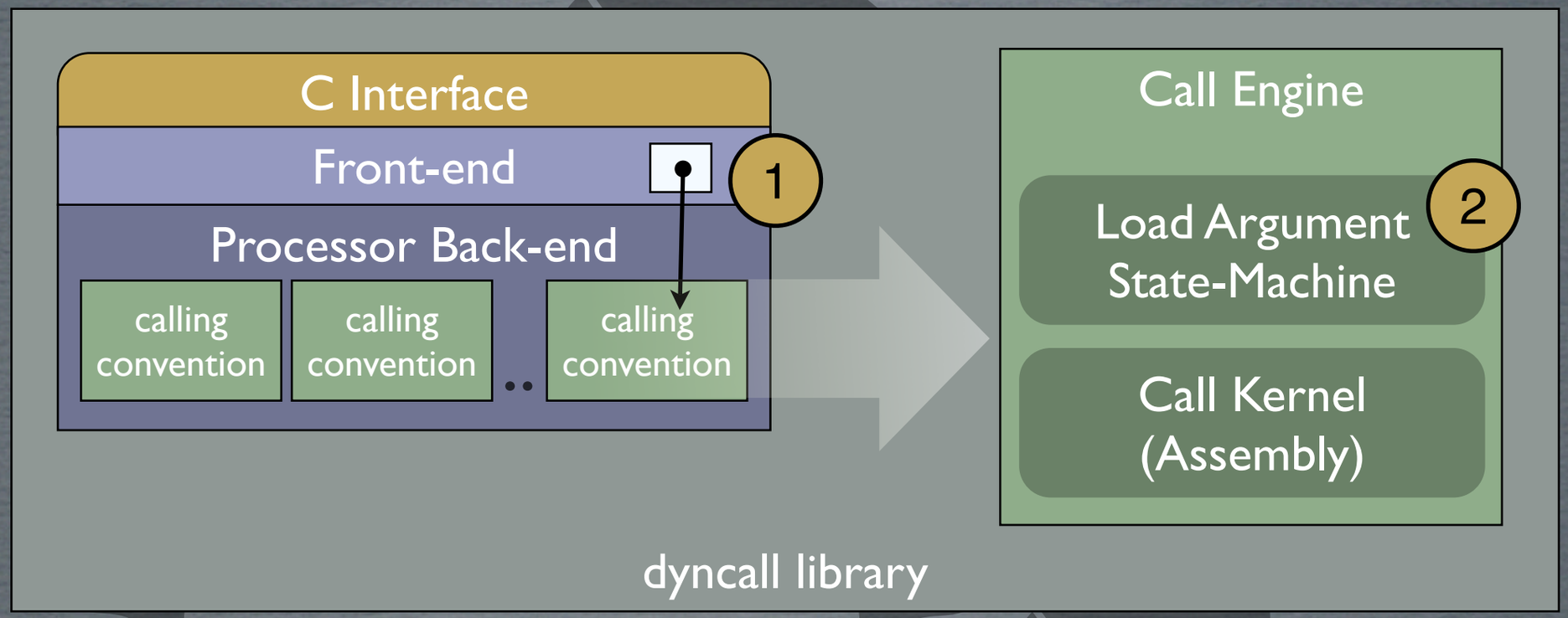
mips

Implementation of the dyncall C library



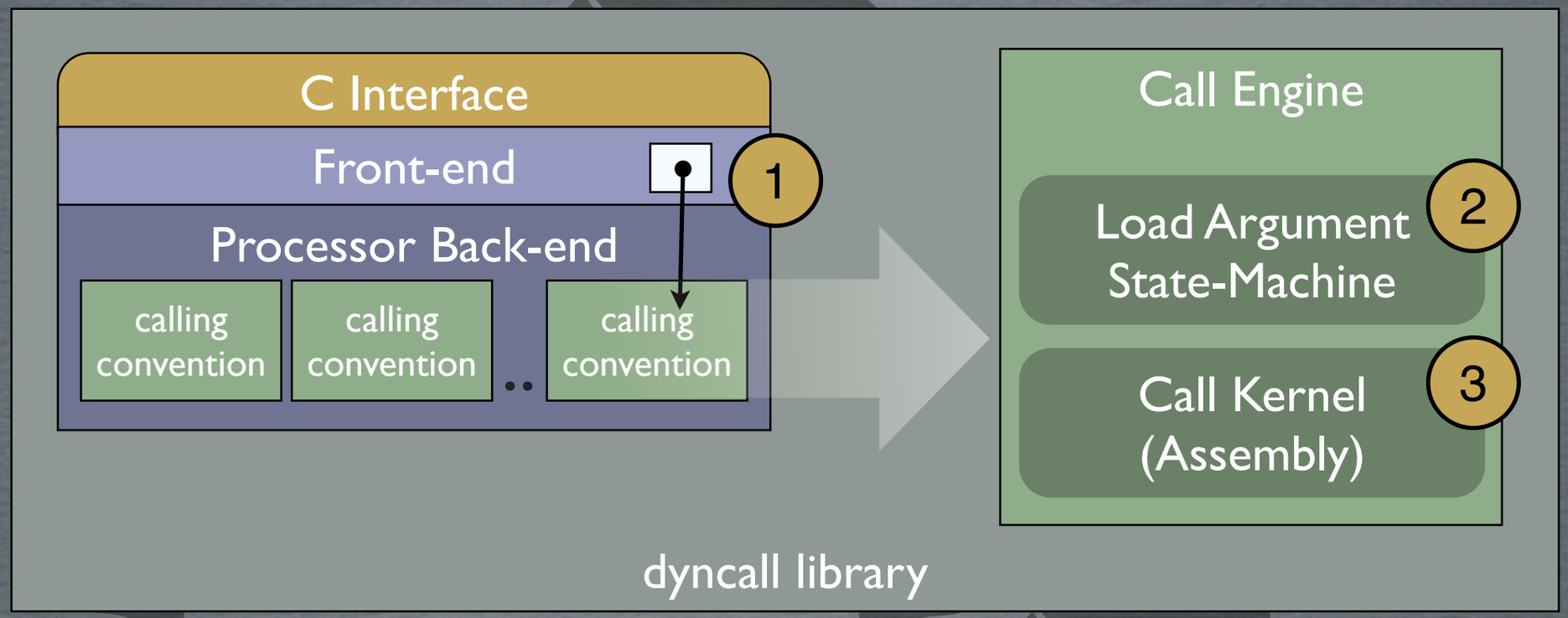
Calling Conventions	cdecl stdcall fastcall.gcc fastcall.ms thiscall.gcc thiscall.ms	win64 system v	darwin system v	arm9e thumb	eabi
C/C++ Compilers			GCC		
Platforms			NetBSD, SOLARIS	NINTENDO DS	SONY
Processors	x86	x64	ppc	arm	mips

Implementation of the dyncall C library



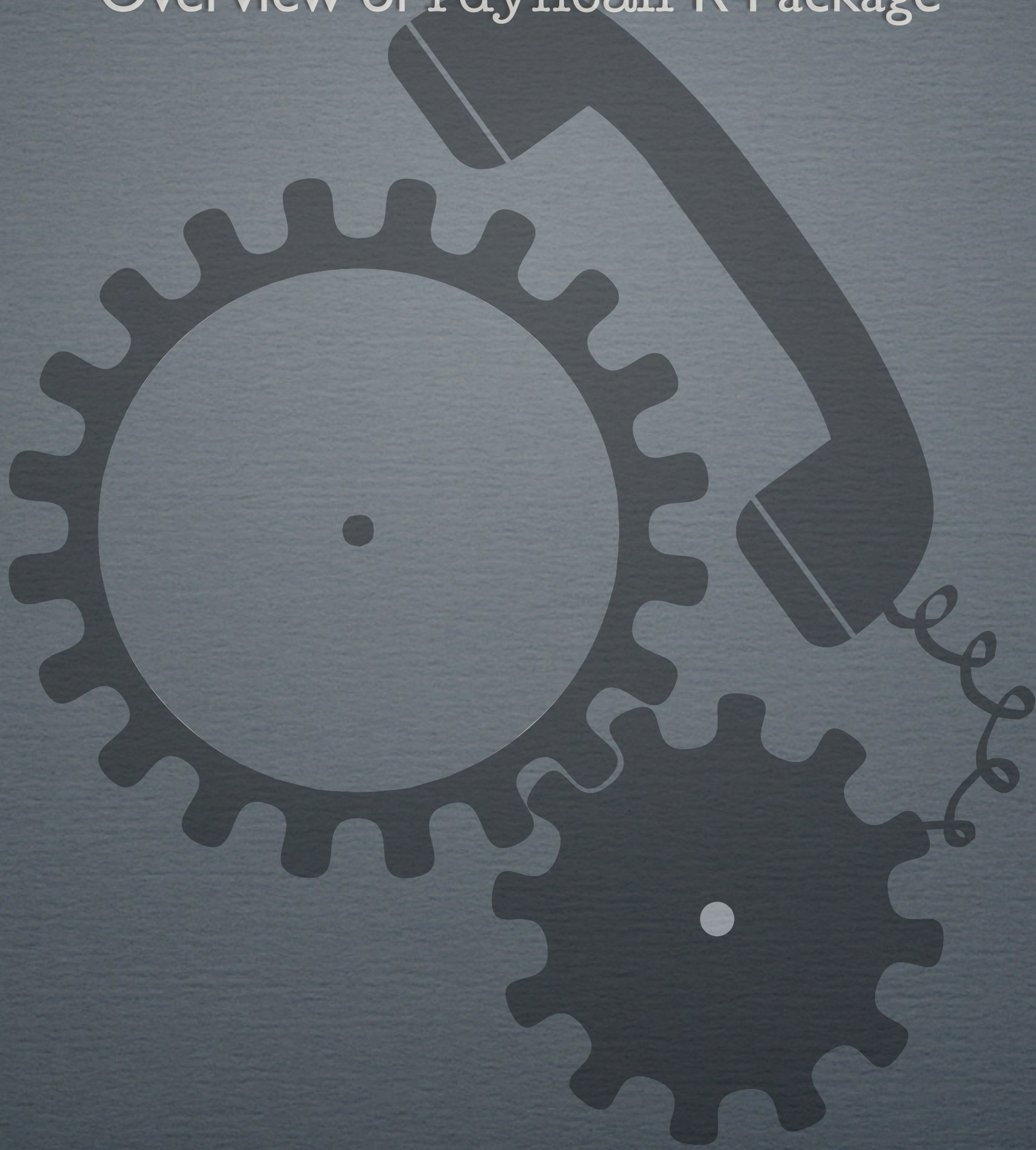
Calling Conventions	cdecl stdcall fastcall.gcc fastcall.ms thiscall.gcc thiscall.ms	win64 system v	darwin system v	arm9e thumb	eabi
C/C++ Compilers			GCC		
Platforms			NetBSD, SOLARIS		
Processors	x86	x64	ppc	arm	mips

Implementation of the dyncall C library



Calling Conventions	cdecl stdcall fastcall.gcc fastcall.ms thiscall.gcc thiscall.ms	win64 system v	darwin system v	arm9e thumb	eabi
C/C++ Compilers			GCC		
Platforms			NetBSD, SOLARIS		
Processors	x86	x64	ppc	arm	mips

Overview of rdyncall R Package



Overview of rdyncall R Package

R implementation

Function Calls

.dyncall(..)

rdyncall C implementation

dyncall

dyncall libraries

Overview of rdyncall R Package

R implementation

Function Calls

.dyncall(..)

Locate/Load Code

dynfind(..)

rdyncall C implementation

dyncall

dynload

dyncall libraries

Overview of rdyncall R Package

R implementation

Function Calls

.dyncall(..)

Locate/Load Code

dynfind(..)

dyncall

```
> dynfind("SDL")
<pointer: 0x1583d670>
attr(,"path")
[1] "/Library/Frameworks/SDL.framework/SDL"
attr(,"auto.unload")
[1] TRUE
```

Overview of rdyncall R Package

R implementation

Function Calls

.dyncall(..)

Locate/Load Code

dynfind(..)

rdyncall C implementation

dyncall

dynload

dyncall libraries

Overview of rdyncall R Package

R implementation

Binding of Libraries / R Wrappers

dynbind(..)

Function Calls

.dyncall(..)

Locate/Load Code

dynfind(..)

rdyncall C implementation

dyncall

dynload

dyncall libraries

Overview of rdyncall R Package

R implementation

Binding of Libraries / R Wrappers

dynbind(..)

Funct

.dynca

d

```
> dynbind("SDL","
SDL_SetVideoMode(iiiI)*<SDL_Surface>;
SDL_GL_SwapBuffers()v;
SDL_PollEvents(*<SDL_Event>)v;SDL_Delay(i)v;")
> ls()
[1] "SDL_SetVideoMode" "SDL_GL_SwapBuffers"
[3] "SDL_PollEvents"   "SDL_Delay"
> SDL_SetVideoMode
function (...)
.dyncall.cdecl(<pointer:0x142b7190>,
"iiiI)*<SDL_Surface>" , ...)
```

Overview of rdyncall R Package

R implementation

Binding of Libraries / R Wrappers

dynbind(..)

Function Calls

.dyncall(..)

Locate/Load Code

dynfind(..)

rdyncall C implementation

dyncall

dynload

dyncall libraries

Overview of rdyncall R Package

R implementation

Binding of Libraries / R Wrappers

dynbind(..)

High-level C Type

new.struct(..)
as.struct(..)

Function Calls

.dyncall(..)

Locate/Load Code

dynfind(..)

Low-level C Type

.pack(..)
.unpack(..)

rdyncall C implementation

dyncall

dynload

dyncall libraries

Overview of rdyncall R Package

R implementation

Binding of Libraries / R Wrappers

dynbind(..)

High-level C Type

new.struct(..)
as.struct(..)

Function Calls

.dyncall(..)

```
> parseStructInfos("SDL_Rect{ssSS}x y w h;")
> x <- new.struct("SDL_Rect")
> x$w <- 100
> typeof(x)
[1] "raw"
> x$w
[1] 100
```

dyncall

dynload

dyncall libraries

Overview of rdyncall R Package

R implementation

Binding of Libraries / R Wrappers

dynbind(..)

High-level C Type

new.struct(..)
as.struct(..)

Function Calls

.dyncall(..)

Locate/Load Code

dynfind(..)

Low-level C Type

.pack(..)
.unpack(..)

rdyncall C implementation

dyncall

dynload

dyncall libraries

Overview of rdyncall R Package

R implementation

Binding of Libraries / R Wrappers

`dynbind(..)`

R Function Callbacks

`new.callback(..)`

High-level C Type

`new.struct(..)`
`as.struct(..)`

Function Calls

`.dyncall(..)`

Locate/Load Code

`dynfind(..)`

Low-level C Type

`.pack(..)`
`.unpack(..)`

rdyncall C implementation

dyncall

dynload

dyncallback

dyncall libraries

Overview of rdyncall R Package

R implementation

Binding of Libraries / R Wrappers

dynbind(..)

R Function Callbacks

High-level C Type

new.struct(..)
as.struct(..)

Function Calls

.dyncall(..)

Locate/Load Code

dynfind(..)

new.callback(..)

Low-level C Type

.pack(..)

```
> tagBegin <- function(handle,tag,attrs){..}
> cb <- new.callback("pZp)v",tagBegin)
> XML_SetElementHandler(handle,cb,NULL)
```

dyncall

dynload

dyncallback

dyncall libraries

Overview of rdyncall R Package

R implementation

Binding of Libraries / R Wrappers

`dynbind(..)`

R Function Callbacks

`new.callback(..)`

High-level C Type

`new.struct(..)`
`as.struct(..)`

Function Calls

`.dyncall(..)`

Locate/Load Code

`dynfind(..)`

Low-level C Type

`.pack(..)`
`.unpack(..)`

rdyncall C implementation

dyncall

dynload

dyncallback

dyncall libraries

Overview of rdyncall R Package

dynport files
Text-based
Binding Meta-Information

R implementation

Dynamic Packages / Multi-Platform Code Bindings

```
dynport(..)
loadDynportNamespace(..)
```

Binding of Libraries / R Wrappers

```
dynbind(..)
```

R Function Callbacks

```
new.callback(..)
```

High-level C Type

```
new.struct(..)
as.struct(..)
```

Function Calls

```
.dyncall(..)
```

Locate/Load Code

```
dynfind(..)
```

Low-level C Type

```
.pack(..)
.unpack(..)
```

rdyncall C implementation

dyncall | dynload | dyncallback

dyncall libraries

Overview of rdyncall R Package

R implementation

Dynamic Packages / Multi-Platform Code Bindings

dynport(..)
loadDynportNamespace(..)

dynport files
Text-based
Binding Meta-Information

Binding of Libraries /

dynbind(..)

```
> dynport(SDL)
> search()
 [1] ".GlobalEnv"         "package:SDL"
 [3] "package:rdyncall" ...
> ls(2)
  [51] "SDLK_F1"              # constants
[539] "SDL_PixelFormat"   # C struct type
[537] "SDL_PauseAudio"    # functions
> unloadNamespace("SDL")
```

Function Calls

.dyncall(..)

Lo

dy

rdyncall C implementation

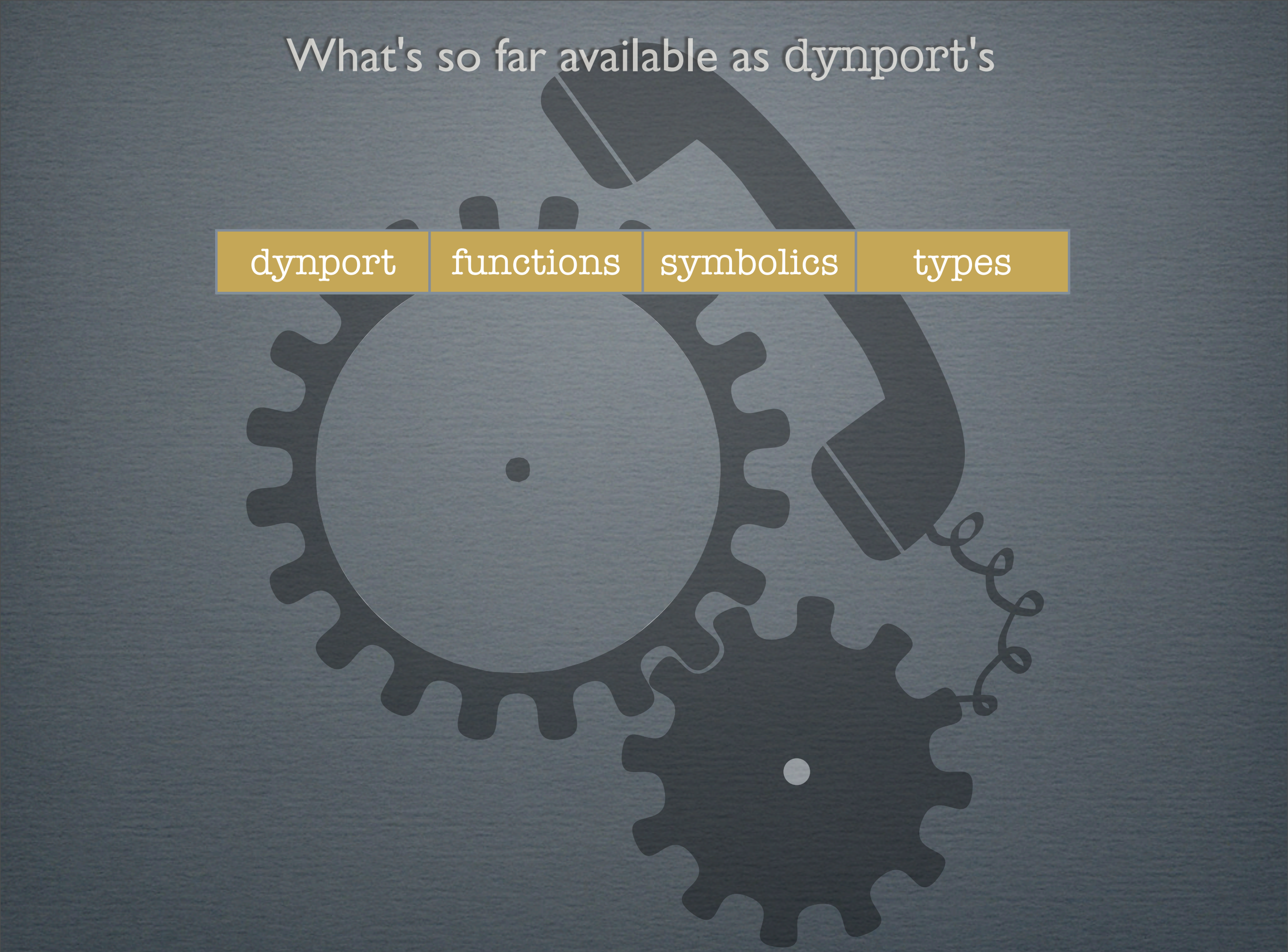
dyncall

dynload

dyncallback

dyncall libraries

What's so far available as dynport's



What's so far available as dynport's

dynport	functions	symbolics	types
SDL	201	416	34
GL	336	3254	-
GLU	59	155	-
glew	1465	-	-
SDL_Image	27	-	-
ode	545	-	NA
expat	65	70	-
R	238	22	NA

the dynport concept

Compiled Libraries

glew

C

OpenGL

C

SDL
Simple Directmedia Layer

C

openAL

C

BULLET
PHYSICS LIBRARY

C

R

C

Dynamic Languages

python™

Perl

Ruby
PROGRAMMING
Language

the dynport concept

Compiled Libraries

glew C

OpenGL C

SDL Simple Directmedia Layer C

OpenAL C

BULLET PHYSICS LIBRARY C

R C

Dynamic Languages

python™

R

Perl

Ruby PROGRAMMING Language

the dynport concept

Compiled Libraries

glew — C

OpenGL — C

SDL (Simple Directmedia Layer) — C

OpenAL — C

BULLET PHYSICS LIBRARY — C

R — C

dynport bindings

Dynamic Languages

python™

R

Perl

Ruby PROGRAMMING Language

the dynport concept

Compiled Libraries

glew

C

OpenGL

C

SDL
Simple Directmedia Layer

C

openAL

C

BULLET
PHYSICS LIBRARY

C

R

C

dynport
bindings

python™

R

Perl

Ruby
PROGRAMMING
Language

Dynamic Languages

the dynport concept

Compiled Libraries

glew

C

OpenGL

C

SDL
Simple Directmedia Layer

C

openAL

C

BULLET
PHYSICS LIBRARY

C

R

C

dynport
bindings

python™

R

Perl

Ruby
PROGRAMMING
Language

Dynamic Languages

the dynport concept

Compiled Libraries

glew

C

OpenGL

C

SDL
Simple Directmedia Layer

C

openAL

C

BULLET
PHYSICS LIBRARY

C

R

C

dynports
repository

dynport
bindings

python™

Perl

R

Dynamic Languages

Ruby
PROGRAMMING
Language

rdyncall and **dyncall**

available open-source (BSD)

R Package available soon on CRAN

http://dyncall.org