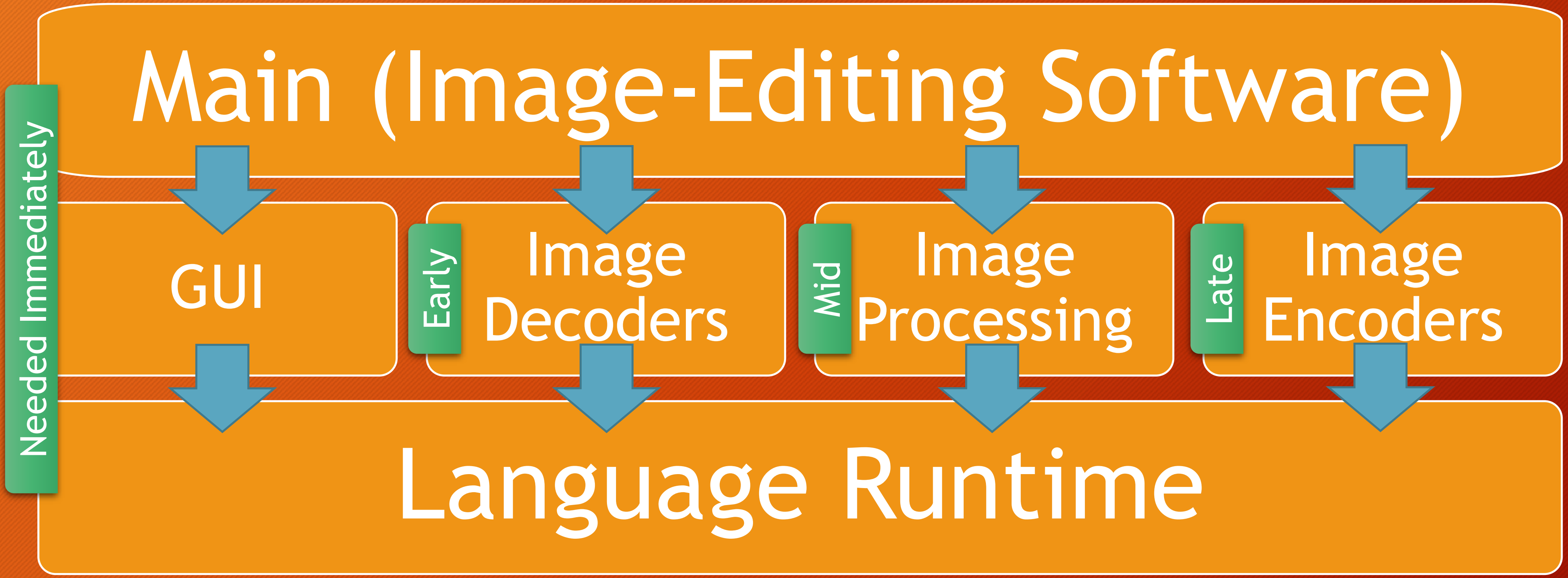
Deferred Loading of Types

Ross Tate

Motivation


Main (Image-Editing Software)
Needed Immediately
GUI
Early
Image Decoders
Mid
Image Processing
Late
Image Encoders
Language Runtime

Scope

- Deferred loading of
  - Functionality (e.g. function definitions)
  - Data (and initialization)
  - Accessors (e.g. field offsets)
  - Types (e.g. struct definitions) ← today’s focus

Example

```
package early;
import later.Bar;

public class Foo {
    private Bar b;
    public Foo() { b = null; }
    public Bar getb() { return this.b; }
    public void setb(Bar b) { this.b = b; }
}
```

- Desire
  - Load Foo early
  - Load Bar later
- Challenge
  - Definition of Foo’s type uses Bar’s type
- Observation
  - Foo’s code does not use Bar’s type’s definition
  - Foo’s code preserves Bar’s invariants

Compilation

```
package early;
import later.Bar;

public class Foo {
    private Bar b;
    public Foo() { b = null; }
    public Bar getb() { return this.b; }
    public void setb(Bar b) { this.b = b; }
}
```

```
(module $Foo
    (type $Bar (import “later” "Bar"))
    (type $Foo (struct
            (field $vtable (ref $FooVTable))
            (field $b (mut (ref null $Bar)))
    ))
    (type $FooVTable (struct
            (field $getb (func (param $this (ref $Foo)) (result (ref null $Bar))))
            (field $setb (func (param $this (ref $Foo)) (param $b (ref null $Bar))))
    ))
    (func (export "__new_Foo") (result (ref $Foo)) (struct.new $Foo (global.get $__Foo_vtable) (ref.null $Bar)))
    (func $Foo.getb (param $this (ref $Foo)) (result (ref null $Bar)) (struct.get $b (local.get $this)))
    (func $Foo.setb (param $this (ref $Foo)) (param $b (ref null $Bar)) (struct.set $b (local.get $this) (local.get $b)))
    (global $__Foo_vtable (ref $FooVTable) (struct.new $FooVTable (ref.func $Foo.getb) (ref.func $Foo.setb)))
    … ;; exports of the relevant field references for the methods $getb and $setb
)
```

Seems to require later.Bar to be loaded

Solution

- Separate type “declaration” and type “definition”
- (declare-type $Bar)
  - Creates a new type $Bar but does not define it
- (define-type $Bar t)
  - Defines declared-but-not-yet-defined $Bar to be type t
- Sound provided every declare-type has at most one define-type
- Loader can declare types for use by “early” modules
  - And “later” modules can establish their definitions when loaded
- Can we compile “early” and “later” modules in the same manner?

Compilation, Revised

```
package early;
import later.Bar;

public class Foo {
    private Bar b;
    public Foo() { b = null; }
    public Bar getb() { return this.b; }
    public void setb(Bar b) { this.b = b; }
}
```

Only change

```
(module $Foo
    (type $Bar (import “later” "Bar"))
    (define-type $Foo (import “early” “Foo”) (struct
            (field $vtable (ref $FooVTable))
            (field $b (mut (ref null $Bar)))
    ))
    (type $FooVTable (struct
            (field $getb (func (param $this (ref $Foo)) (result (ref null $Bar))))
            (field $setb (func (param $this (ref $Foo)) (param $b (ref null $Bar))))
    ))
    (func (export "__new_Foo") (result (ref $Foo)) (struct.new $Foo (global.get $__Foo_vtable) (ref.null $Bar)))
    (func $Foo.getb (param $this (ref $Foo)) (result (ref null $Bar)) (struct.get $b (local.get $this)))
    (func $Foo.setb (param $this (ref $Foo)) (param $b (ref null $Bar)) (struct.set $b (local.get $this) (local.get $b)))
    (global $__Foo_vtable (ref $FooVTable) (struct.new $FooVTable (ref.func $Foo.getb) (ref.func $Foo.setb)))
    … ;; exports of the relevant field references for the methods $getb and $setb
)
```

Early Loading and Static Linking:

```
(module $__glue
    (declare-type $Foo)
    (declare-type $Bar)
    (instance $Foo (import “later” "Bar" (type $Bar))
                   (import “early” "Foo" (type $Foo)))
    (instance $Bar (import “later” "Bar" (type $Bar)))
    …
)
```

Bonus: Mutual Recursion (without Inversion)

```
package later;
import early.Foo;

public class Bar {
    private Foo f;
    public Bar() { f = null; }
    public Foo getf() { return this.f; }
    public void setf(Foo f) { this.f = f; }
}
```

Same as Foo (modulo renaming)

```
(module $Bar
    (type $Foo (import “early” “Foo"))
    (define-type $Bar (import “later” “Bar”)) (struct
            (field $vtable (ref $BarVTable))
            (field $f (mut (ref null $Foo)))
    ))
    (type $BarVTable (struct
            (field $getf (func (param $this (ref $Bar)) (result (ref null $Foo))))
            (field $setf (func (param $this (ref $Bar)) (param $f (ref null $Foo))))
    ))
    (func (export "__new_Bar") (result (ref $Bar)) (struct.new $Bar (global.get $__Bar_vtable) (ref.null $Foo)))
    (func $Bar.getf (param $this (ref $Bar)) (result (ref null $Foo)) (struct.get $f (local.get $this)))
    (func $Bar.setf (param $this (ref $Bar)) (param $f (ref null $Foo)) (struct.set $f (local.get $this) (local.get $f)))
    (global $__Bar_vtable (ref $BarVTable) (struct.new $BarVTable (ref.func $Bar.getf) (ref.func $Bar.setf)))
    … ;; exports of the relevant field references for the methods $getf and $setf
)
```

Early Loading and Static Linking:

```
(module $__glue
    (declare-type $Foo)
    (declare-type $Bar)
    (instance $Foo (import “later” "Bar" (type $Bar))
                   (import “early” "Foo" (type $Foo)))
    (instance $Bar (import “early” “Foo” (type $Foo))
                   (import “later” "Bar" (type $Bar)))
…)
```