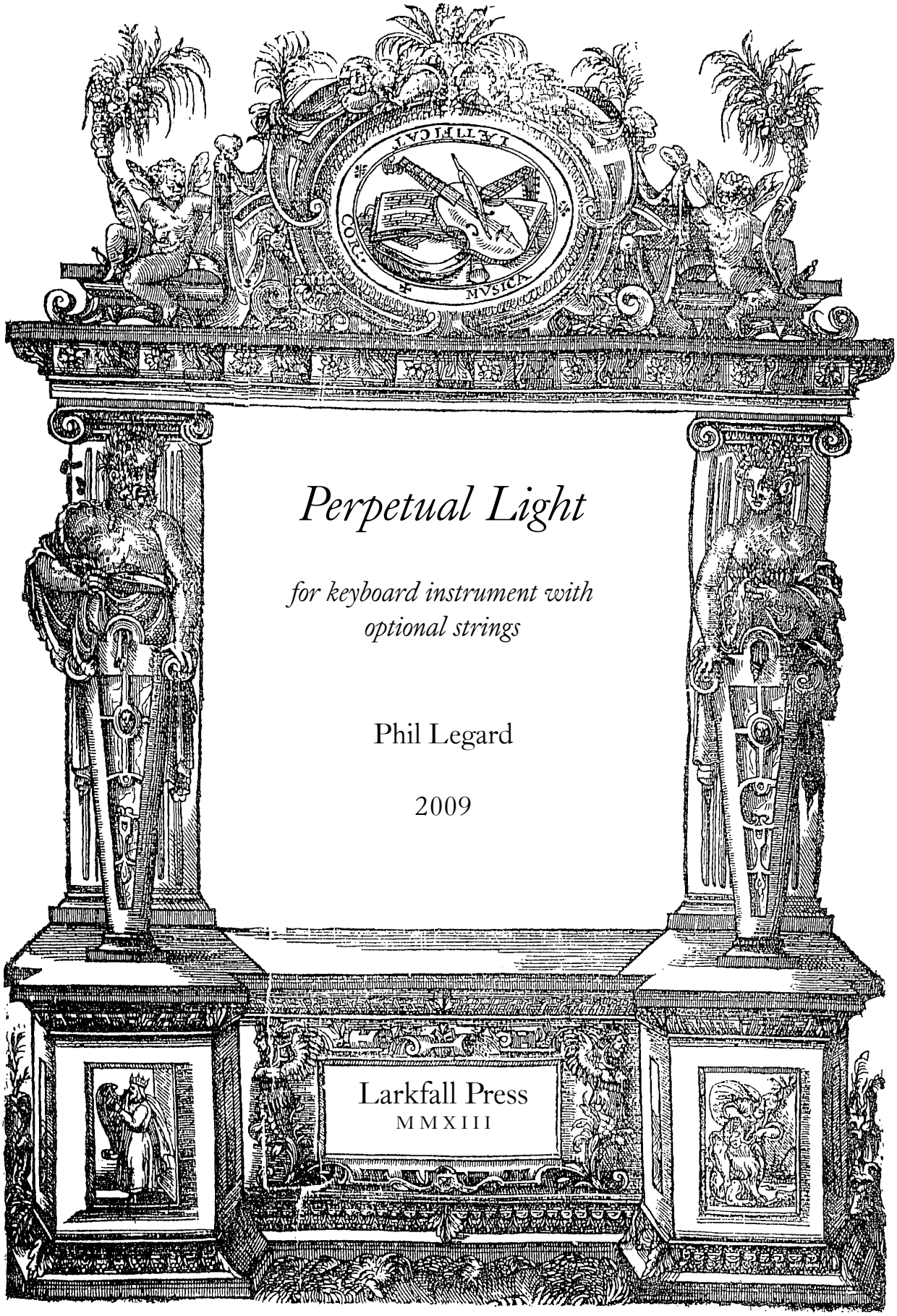


Perpetual Light

for keyboard instrument with optional strings

Phil Legard

2009

Larkfall Press
MMXIII

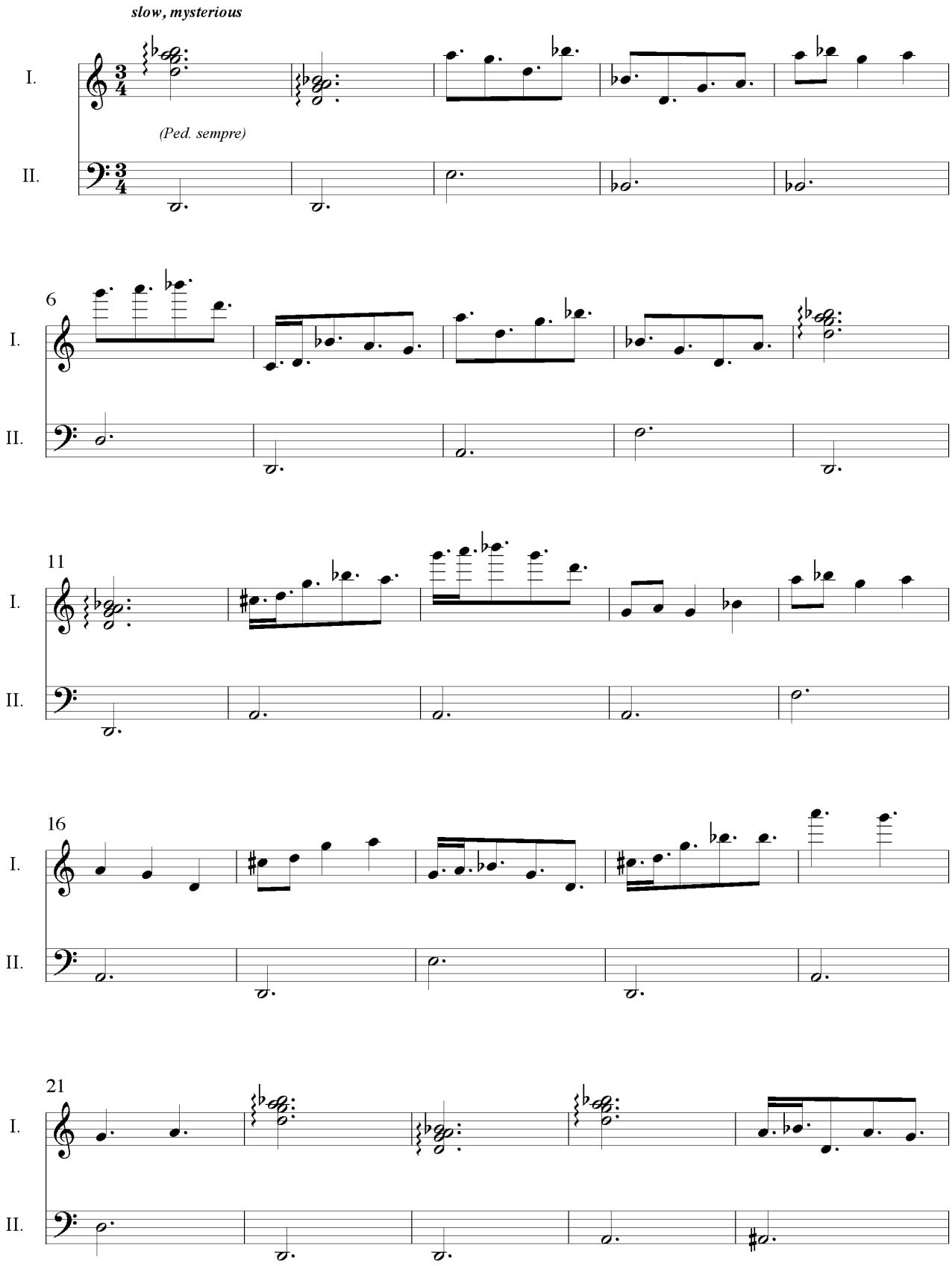
Note

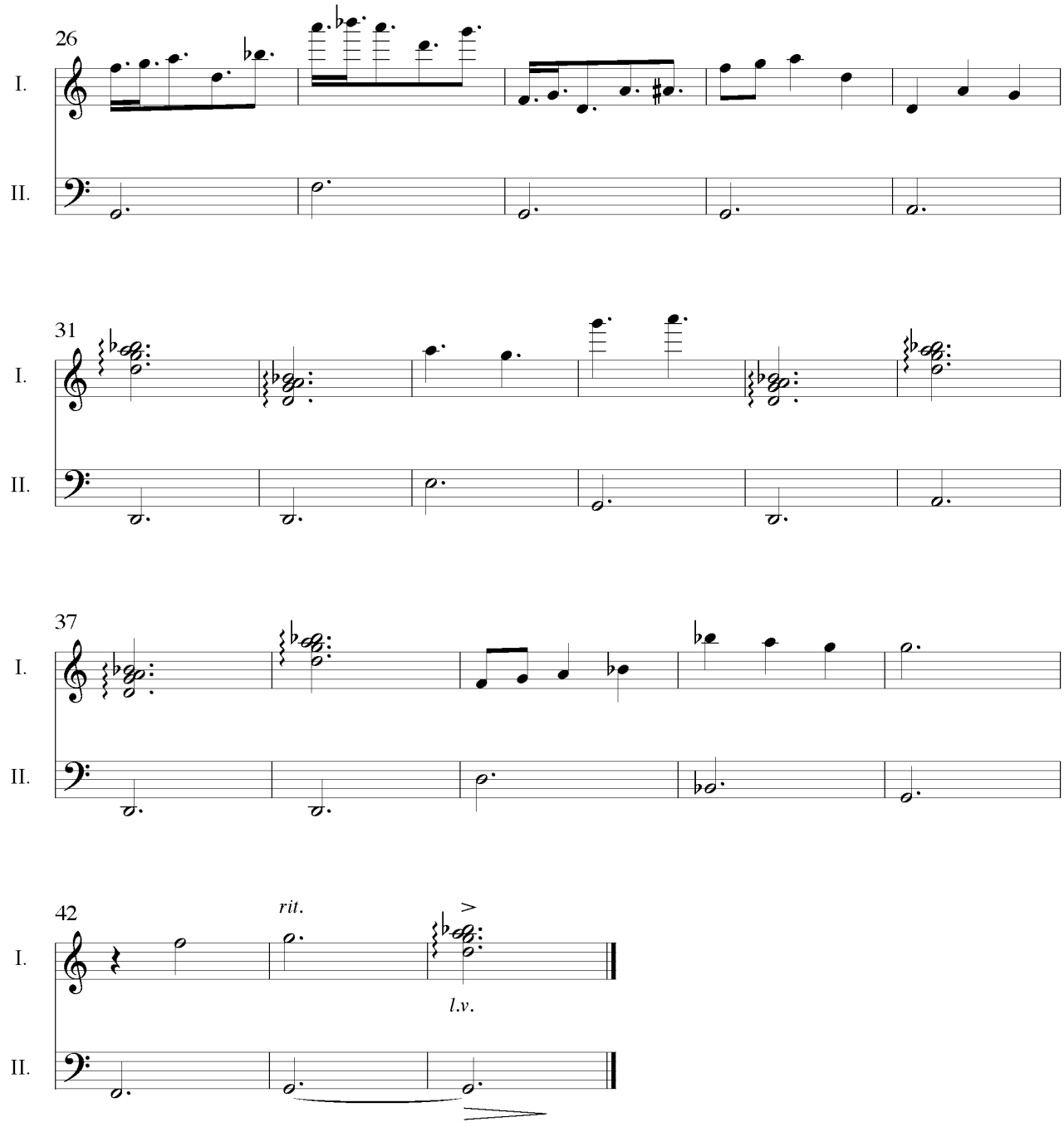
Perpetual Light was written for keyboard instrument – ideally an electric piano. The staves marked I and II correspond to the right and left hands of a traditional grand stave. Staff II may also be doubled with a string instrument – transposing the material by octaves as appropriate.

Perpetual Light was realised in code using the Symbolic Composer language. The source code for the piece is appended to this score.

Perpetual Light
for keyboard instrument with optional strings

Phil Legard




26
I.
II.
31
I.
II.
37
I.
II.
42
rit.
l.v.
I.
II.

```
; Perpetual Light: Symbolic Composer code
; Phil Legard, 2009

(init-rnd 0.12)

(setq chd1 '((adef)))

(setq S1 (symbol-repeat 22 chd1))

(setq R1 (symbol-repeat 22 '((1/2.))))

(setq templ (append '(= =) (gen-template nil 1 1 20)))

(setq S2 (do-section templ '(symbol-melodize x) S1))
(setq S3 (do-section templ '(symbol-shuffle x) S2))
(setq S4 (do-section templ '(symbol-trim (pick1 nil '(2 3 4)) x) S3))

(setq templ2 (append '(= =) (gen-template nil 4 1 20)))
(setq S5 (flatten-sublist (do-section templ2
          '(symbol-divide (length x) nil nil
           (append x (symbol-retrograde x))) S4)))

(setq R2 (do-section :all '(length-repeat-l (length x) '(1/2.)) S4))

(setq R3 (flatten-sublist (do-section templ2
               '(symbol-divide (length x) nil nil (append x x)) R2)))

(def-neuron templNeuron
    (in 1 '1) '(=)
    (in 1 '2) '(=)
    (in 1 '3) '(x)
    (in 1 '4) '(x)
    (otherwise '(=))
)

(setq templ3 (do-section :all '(length x) S5))

(setq templ4 (run-neuron 'templNeuron templ3))
(setq templ5 (flatten (do-section templ4
               '(pick1 nil '(= x)) (symbol-divide 1 nil nil templ4))))

(setq S6 (do-section templ5
                 '(append (symbol-transpose -1 (list (car x))) x) S5))

(setq R6 (do-section templ5 '(append (change-length :divide 2
         (list (car x)) :ratio) (change-length :divide 2
         (list (car x)) :ratio)  (cdr x)) R3))

(setq S5b (do-section :all '(symbol-melodize x) S5))
(setq S6b (do-section :all '(symbol-shuffle x) S5b))
(setq S7b (symbol-divide 1 nil nil (do-section :all '(car x) S5b)))
(setq S8b (append '((= a)) (cdr S7b)))
(setq R3b (symbol-repeat (length S7b) '((1/2.))))
(setq R4b (append '((1/8 1/2.)) (cdr R3b)))

(def-tonality
  v1 (activate-tonality (harmonic-minor d 6) (natural-minor d 5)
     (harmonic-minor d 6) (natural-minor d 5)(harmonic-minor d 6)
```

```
      (natural-minor d 7) (natural-minor d 5))
  v2 (activate-tonality (harmonic-minor d 3) (natural-minor d 3)
      (harmonic-minor a 3))
)

(def-symbol
  v1 S6
  v2 S8b
)

(def-length
 v1 R6
 v2 R4b
)

(def-zone
 v1 (make-zone (get-lengths-of 'v1) :ratio)
 v2 (make-zone (get-lengths-of 'v2) :ratio)
)

(def-velocity
 v1 (vector-round 30 40 (gen-noise-white (length (flatten S5))))
 v2 (vector-round 30 40 (gen-noise-white (length (flatten S8b))))
)

(def-channel
 v1 1
 v2 2
)

(def-program gm-sound-set
 v1 bright-acoustic-piano
 v2 bright-acoustic-piano
)

(def-tempo 62)

(compile-instrument-p "ccl;output:" "perpetual_light"
   v1
   v2
)
```