A Tangram Puzzle Solver in Common Lisp

Michael Wessel

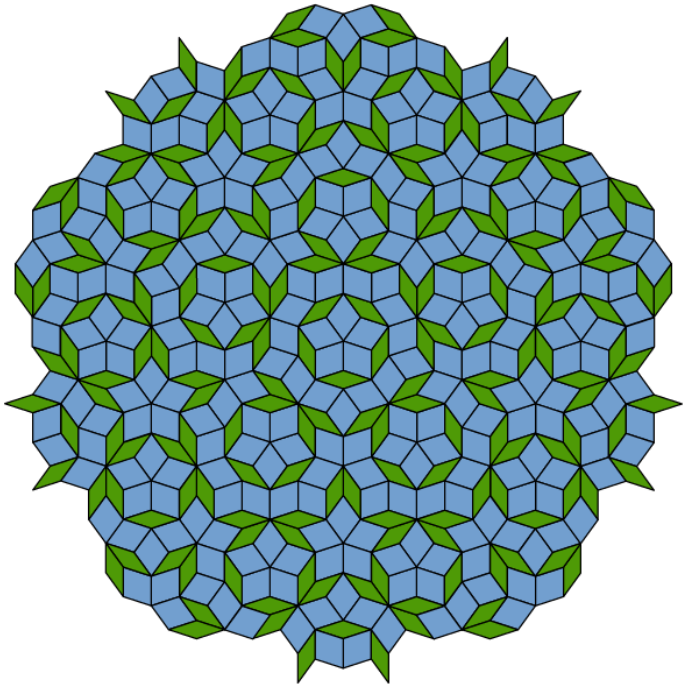
May 3rd 2021

ELS‘21

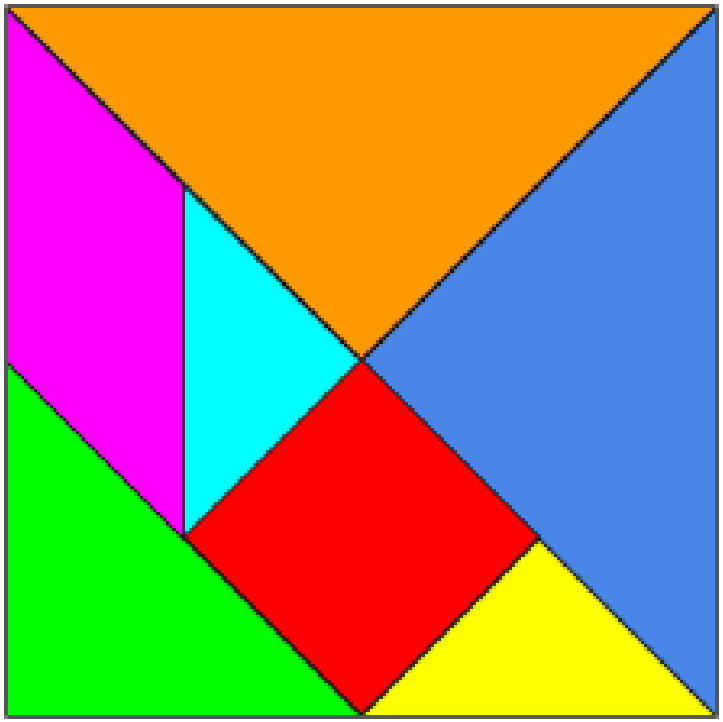
Talk Outline

- **Disclaimer**: this is a demo talk
  - Video of program in action & some Lisp code
  - No deep theoretical insights, but some practical insights
  - This is a fun recreational spare time project
  - Not associated with my employer in any way
- The program was written in 2003, reactivated now
  - To get back to Lisp – **Q: Would I still be able to maintain and extent my Lisp code from almost 20 years ago?**
  - To **finish what was started** – the original program had some deficiencies & bugs
  - To **catch up with the Lisp community**
    (paper motivation – last time I contributed was 15 years ago)
  - Because of a **recent book that really fascinated me** (see at the end)

Tiling Problems are Fascinating!
Penrose Tiling & Tangram Set



Aperiodic Penrose Tiling
(Extends Infinitely)
Source : Wikipedia CC

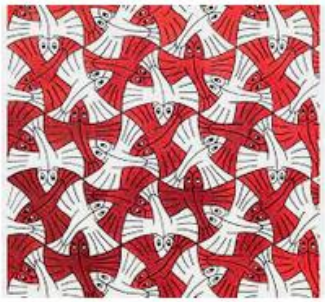


Tangram Tiling
(Finite)
Source: Wikipedia CC

Google tiling escher

Dog Pattern | Schnauzer | M. C. E...
pinterest.com

How to make an Escher tiling
static1.squarespace.com

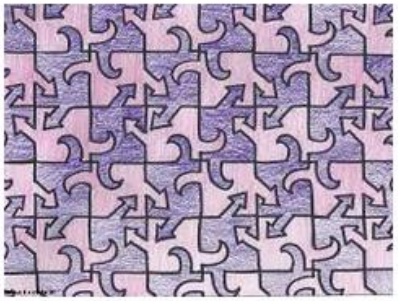


Escher tessellations ...
pinterest.com

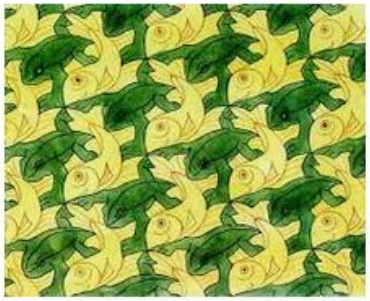
Tessellation patterns ...
pinterest.com

Tessellation patterns ...
pinterest.nz

Tessellation Patterns - From ...
widewalls.ch



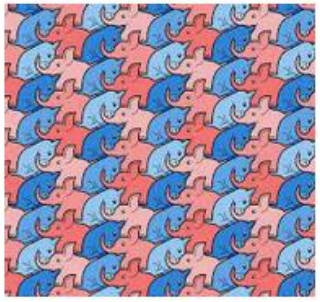
Tessellation art ...
pinterest.com



Tessellations - The Curious Question...
sites.google.com

Tessellation Patterns - From ...
widewalls.ch

Seahorse Tessellation by ...
pinterest.ca



Elephant and Donkey Tessellati...
pinterest.com

Pin on tessellations
pinterest.com

mc escher ...
pinterest.com

330 Tessellations ideas | art...
pinterest.com

tangram - Google Search

https://www.google.com/search?q=tangram&tbm=isch&ved=2ahUKEwj

Legen Sie Ihre Lesezeichen hier in der Lesezeichen-Symbolleiste ab, um schnell darauf zuzugreifen. Lesezeichen verwalten...

Weitere Lesezeichen

Google

tangram

Ads · Shop tangram

Jumbo Magnetic QuietShape Foam

$21.95

EAI Education

Tangram Shapes - Download Digital

$6.70

Etsy

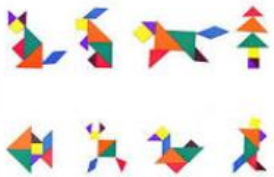
Free shipping

Tangram Cats - 16 Quilt Block Patterns

$16.25

Etsy

Free shipping



Colorful Wooden Tangram Brain

$3.88

eBay

Free shipping

Wooden Tangrams Puzzles, 28 Pieces

$9.99

Amazon.com

Tangram S Sticker | R

$2.50

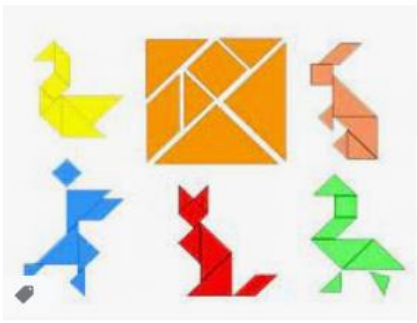
Redbubble

Tangram - Wikipedia

en.wikipedia.org

Newcreativetop 7 Piece Children ...

amazon.com



tangram puzzle in wood - Logica Puzz...

logicagiochi.com

Wooden Tangrams Puzzles, 28 Pie...

amazon.com

tangram bat puzzle - Google S

https://www.google.com/search?q=tangram+bat+puzzle&client=firefox

Legen Sie Ihre Lesezeichen hier in der Lesezeichen-Symbolleiste ab, um schnell darauf zuzugreifen. Lesezeichen verwalten...

Weitere Lesezeichen

Google

tangram bat puzzle

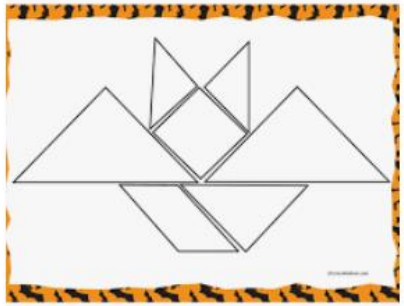
All | Shopping | Images | Videos | News | More | Settings | Tools | Collections

halloween | wall art | 2d shapes | white background | math | pumpkin

Color tangram puzzle in bat shape ...
pixtastock.com

Tangram Bat Shape and So...
supercoloring.com

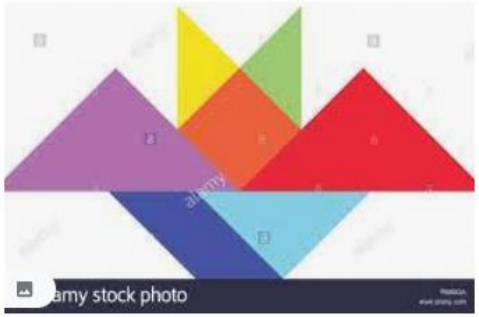
Color Wood Tangram Puzzle In Flying...
123rf.com



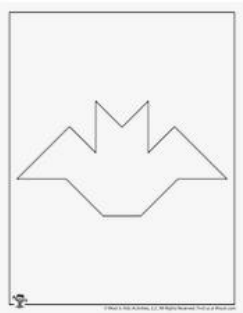
Bat Tangram Puzzle Puzzles Handma...
alsgroup.cl

Make A Tangram Bat | Tangram, Tangram ...
pinterest.com

Tangram printable ...
pinterest.com

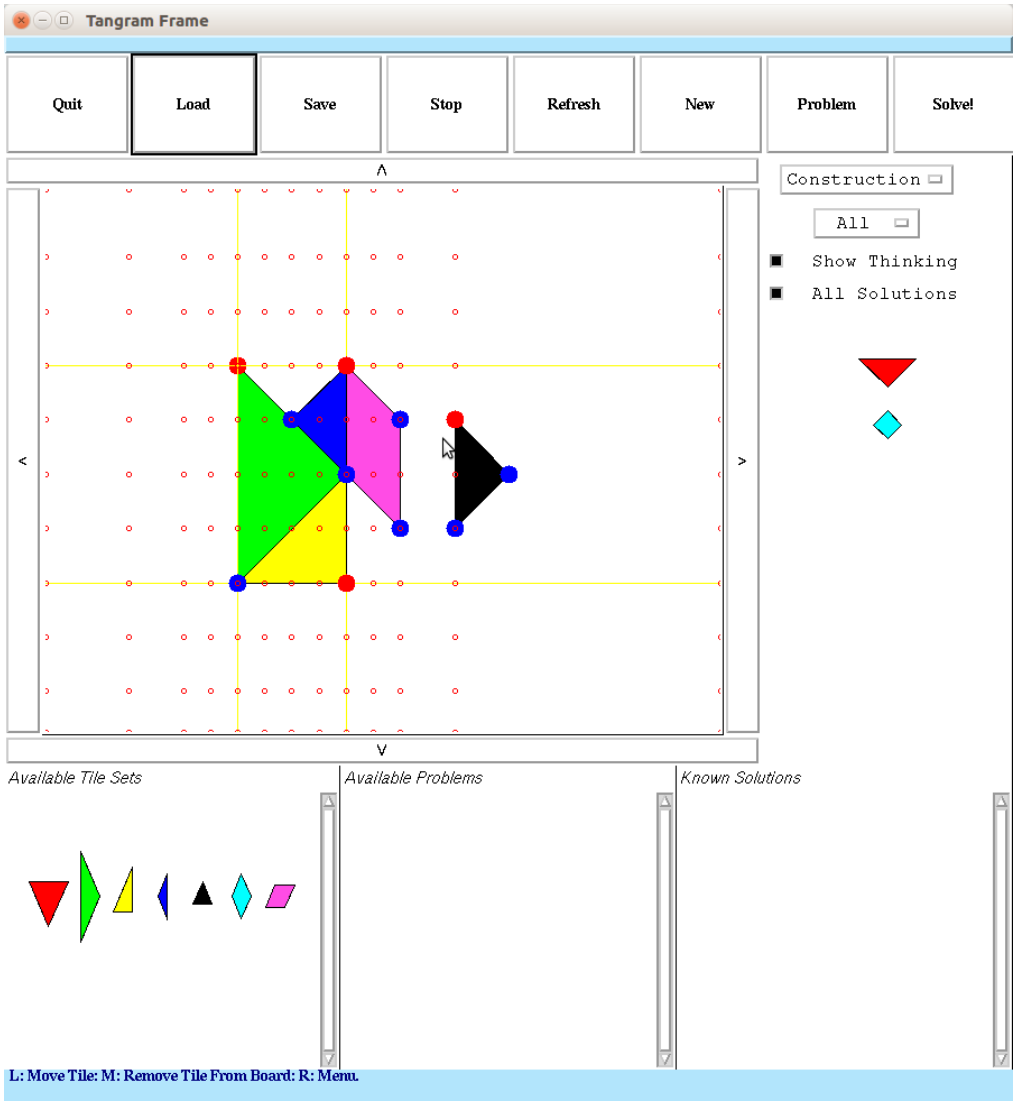


Color tangram puzzle in bat shape on ...
alamy.com



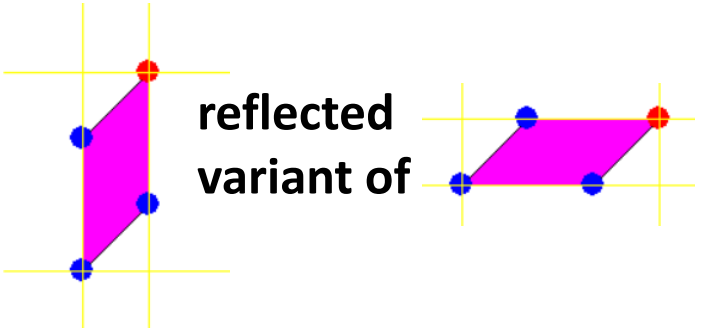
Bat Halloween Tangra...
woojr.com

Editor Demo Video

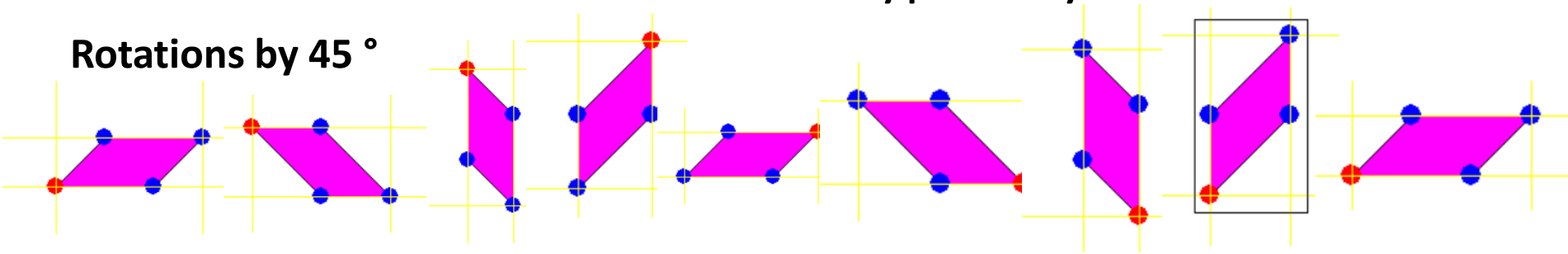


Rules of the Tangram Game

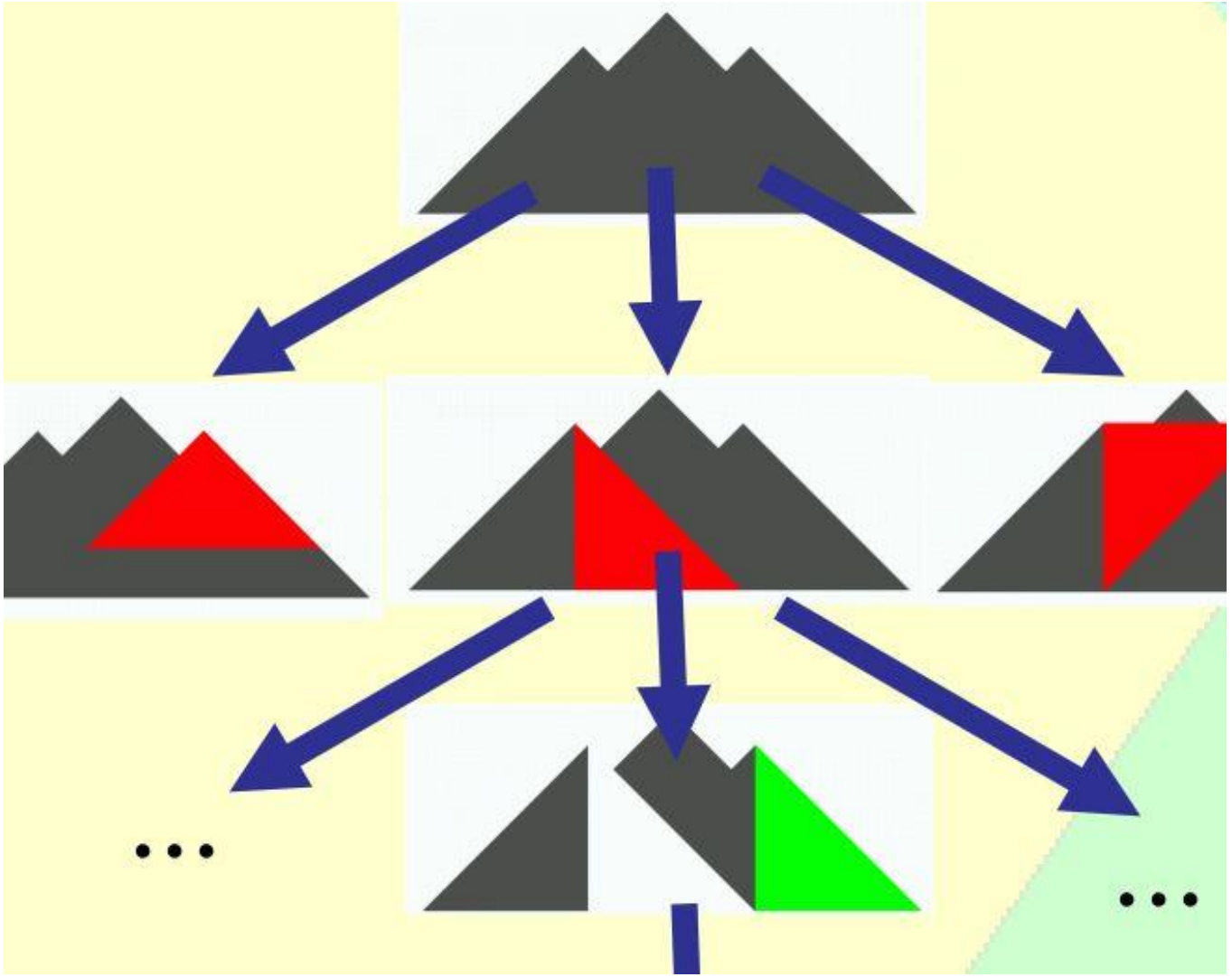
- Each tile must be used EXACTLY once
- The silhoutte / outline polygon must be partioned completely
  - no remaining space
  - no overlaps between tiles
- Tiles can be translated, rotated and reflected (flipped over)
- The reflected variants of each tile (modulo congruency) constitute a **TileType**; only ONE reflected variant of each TileType may be used



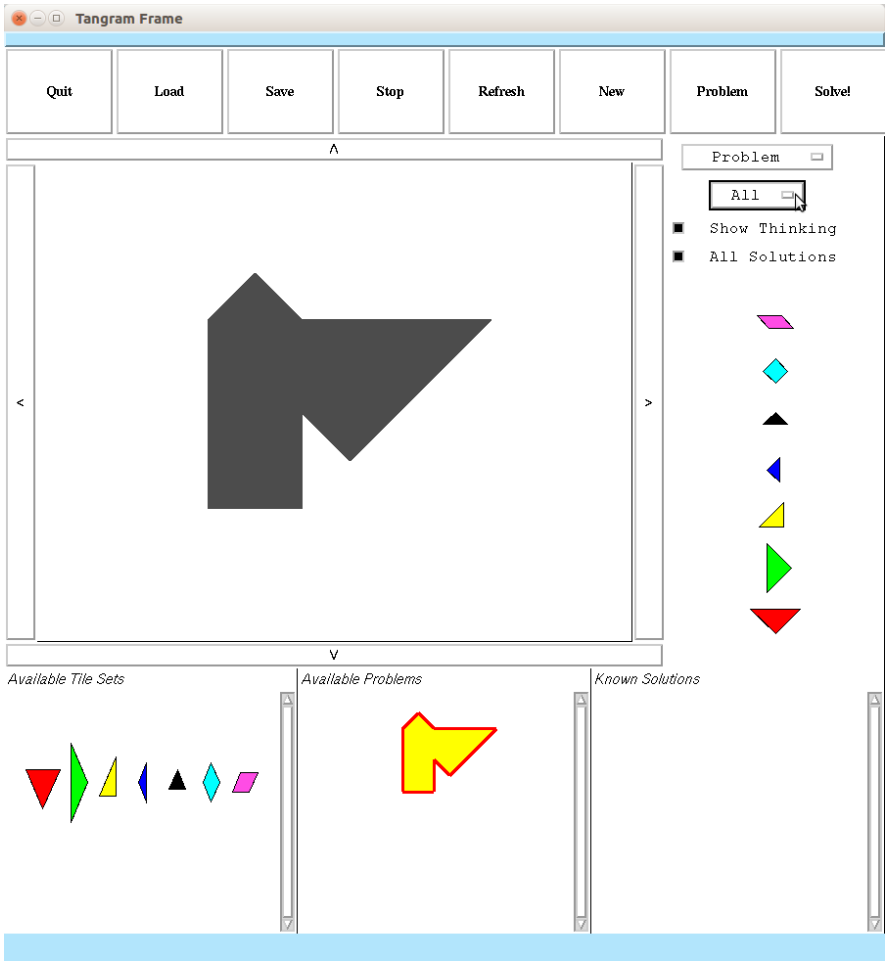
Rotations by 45 °

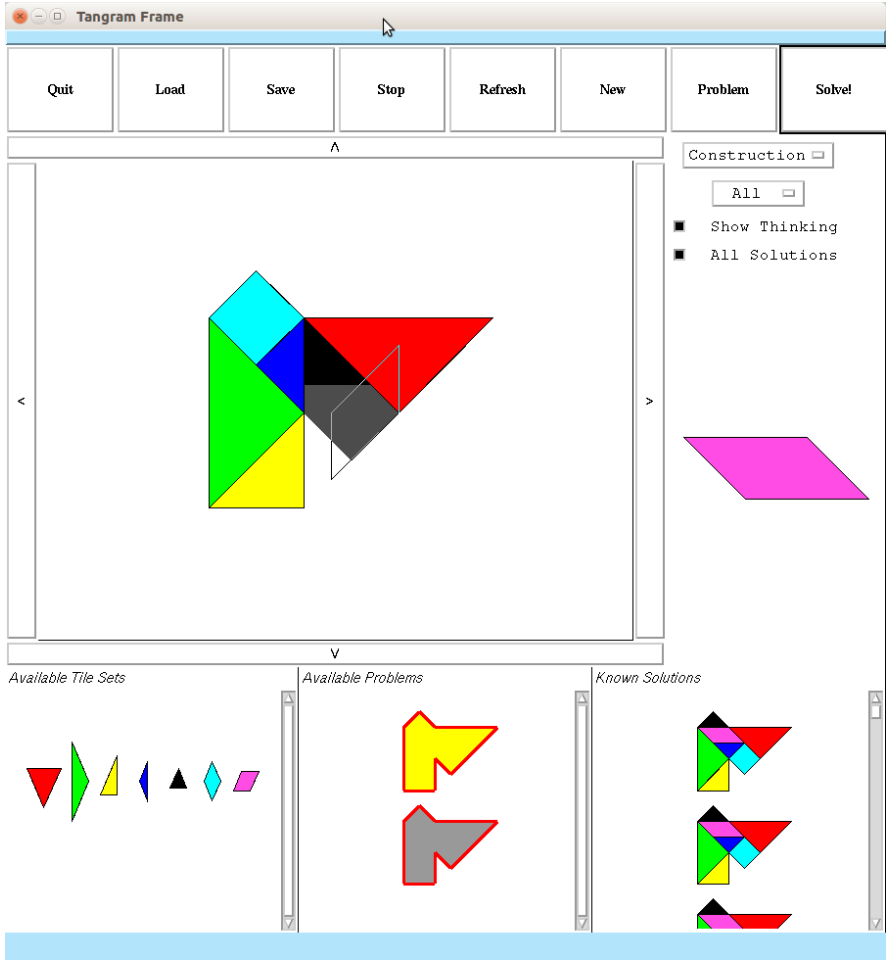


Solving Tiling Problems by Geometric Search



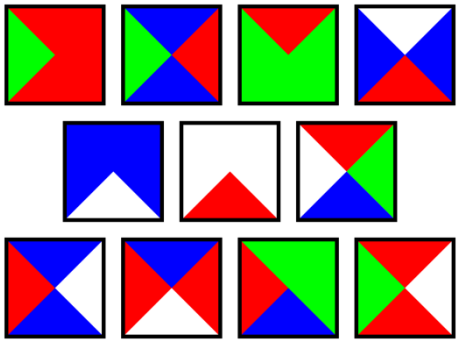
Solver Demo Video



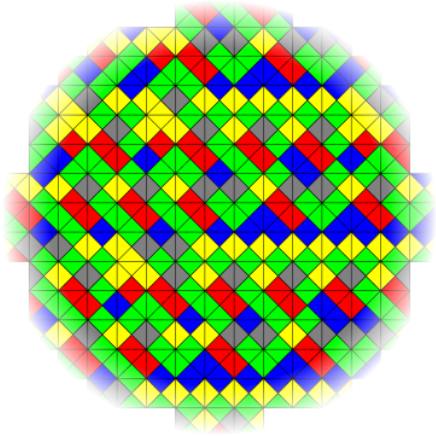


Comments on Problem Complexity

- It is undecidable if an arbitary set of (Wang domino) tiles tiles the (infinte) plane
  - reduction to Halting Problem of Turing Machines
- Wang 1961:
  - Conjecture: if a set of finite tiles tiles the (infinite) plane, then there also exists a **periodic** tiling
  - If the Conjecture is true, then it implies the existence of a decision algorithm
- Berger 1966 (Wang‘s student):
  - That algorithm does not exist – hence, the conjecture is wrong
  - What‘s wrong with it? There also exist **aperiodic** tilings
  - See 11 Wang Dominos (Jeandel and Rao 2015)
  - Penrose tilings
- Tangram is NP-hard
- Certain variants are NP-complete (e.g., symmetric puzzles)



Wang Dominos (Aperiodic)
Source: Wikipedia CC



Wang Tesselation with 13 Tiles
Source: Wikipedia CC

Search Function – 1 / 3

```
(defun find-covering (polygon-set tile-types &key (remove-used-tiles-p t)
                                  frame
                                  all-solutions-p
                                  allow-less-tiles-p
                                  show-thinking-fn)

  (let ((solutions nil))

    (labels ((goal-p (polygons tile-types)
               (and (not polygons)
                    (or allow-less-tiles-p
                        (not tile-types))))

             (do-it (polygons tile-types history)

               (incf *inc-counter*)

               (cond ((goal-p polygons tile-types)

                      (push
                       (list polygons tile-types history)
                       solutions)

                      (when show-thinking-fn
                        (funcall show-thinking-fn polygons tile-types history t))

                      (unless all-solutions-p
                        (return-from find-covering solutions)))
```

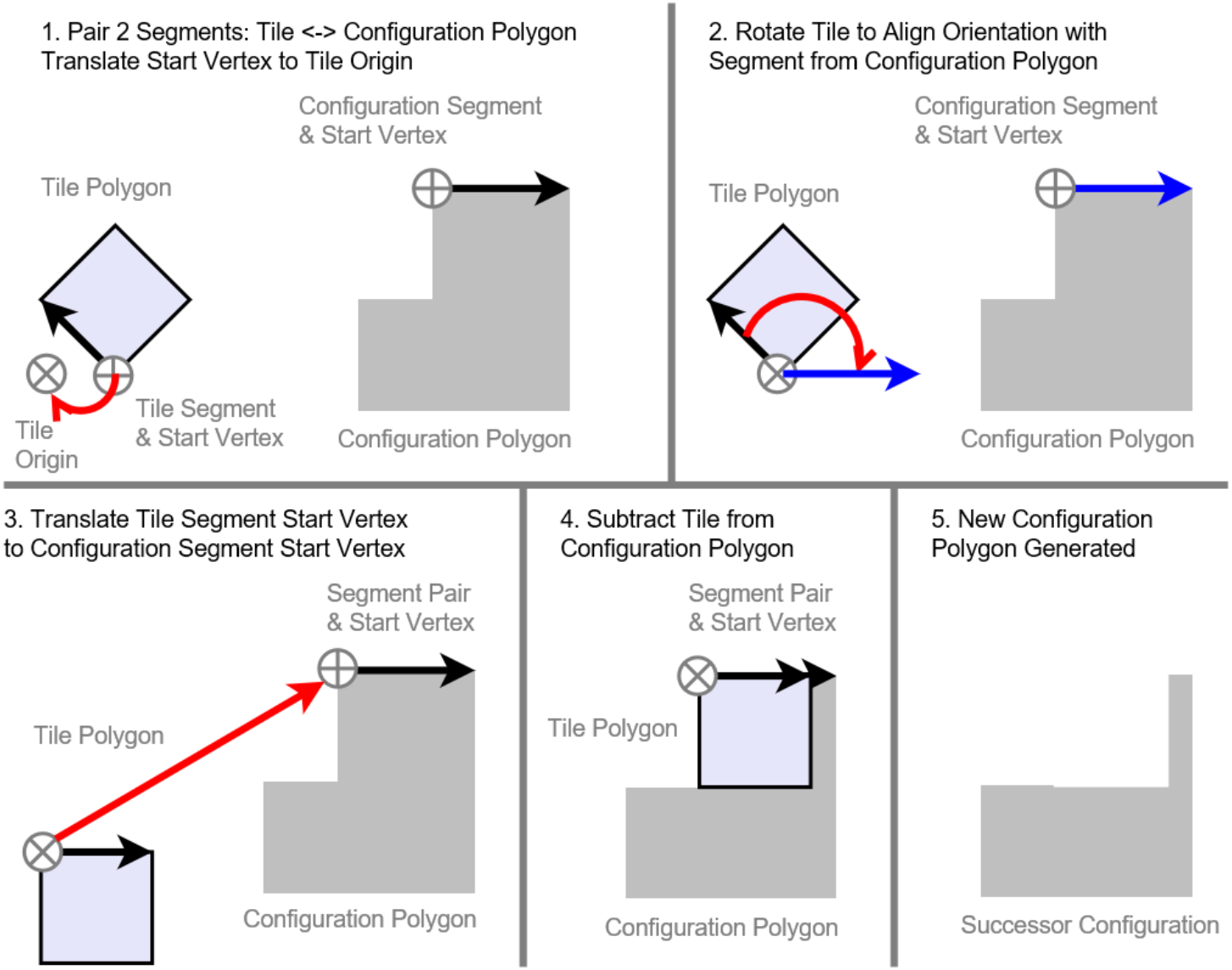
Search Function – 2 / 3

```
(t

 (when show-thinking-fn (funcall show-thinking-fn polygons tile-types history nil))

 #+:ignore
 (sleep 1)

 (let* ((alignments nil)
        (perfect-alignments nil))

   (dolist (polygon polygons)
     (dolist (tile-type tile-types)
       ;; tile-type is the set of reflected variants of that tile
       (let ((rem-tile-types
              (if remove-used-tiles-p
                  (remove tile-type tile-types)
                tile-types)))
         (dolist (tile tile-type)
           (dolist (succ-conf
                    (find-possible-alignments polygon (shape tile)
                                              :frame frame
                                              :color (color tile)
                                              :score-p t
                                              :rem-tile-types rem-tile-types))
             ;; succ-conf = (list h aligned-poly succ-conf orig-poly rem-tile-types)
             (if (eq (third succ-conf) :match)
                 (push (list succ-conf tile ) perfect-alignments)
               (push (list succ-conf tile ) alignments))))))))


   (let ((sorted (append (sort perfect-alignments #'tuple-> :key #'caar)
                         (subseq (sort alignments #'tuple-> :key #'caar)
                                 ;; best n only ?
                                 0 (length alignments)))))
```

Search Function – 3 / 3

```
(let ((sorted (append (sort perfect-alignments #'tuple-> :key #'caar)
                      (subseq (sort alignments #'tuple-> :key #'caar)
                              ;; best n only ?
                              0 (length alignments)))))

  (push (list *inc-counter* (length tile-types) (length sorted)) *statistics*)

  (loop as ((h aligned-poly succ-conf orig-poly rem-tile-types) tile) in sorted do

        (let* ((rem-polygons (remove orig-poly polygons)))
          (do-it (if (eq succ-conf :match)
                     rem-polygons
                   (append succ-conf rem-polygons))
                 rem-tile-types
                 (cons (list aligned-poly tile) history)))))))))

(do-it polygon-set tile-types nil)

solutions)))
```

Successor Generator Function

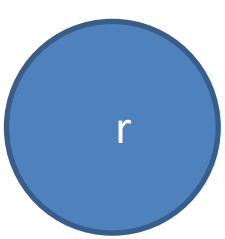


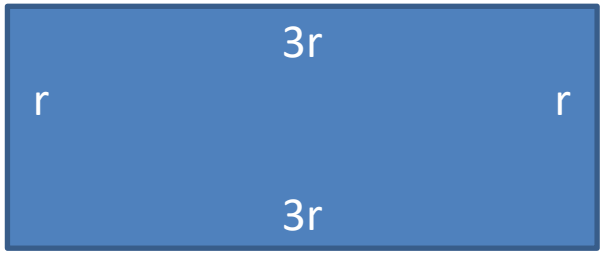
Geometric Heuristics

- **Tile area** (biggest first)
- **Compactness** of successor polygon *P*

$$compactness(P) = \frac{circumference(P)}{\sqrt{area(P)}}$$

```
(defun compactness (poly)
  (with-no-matrix-at-all
    (/
     (loop as s in (segments poly)
           sum (length-of-line s))
     (sqrt (calculate-area poly)))))
```

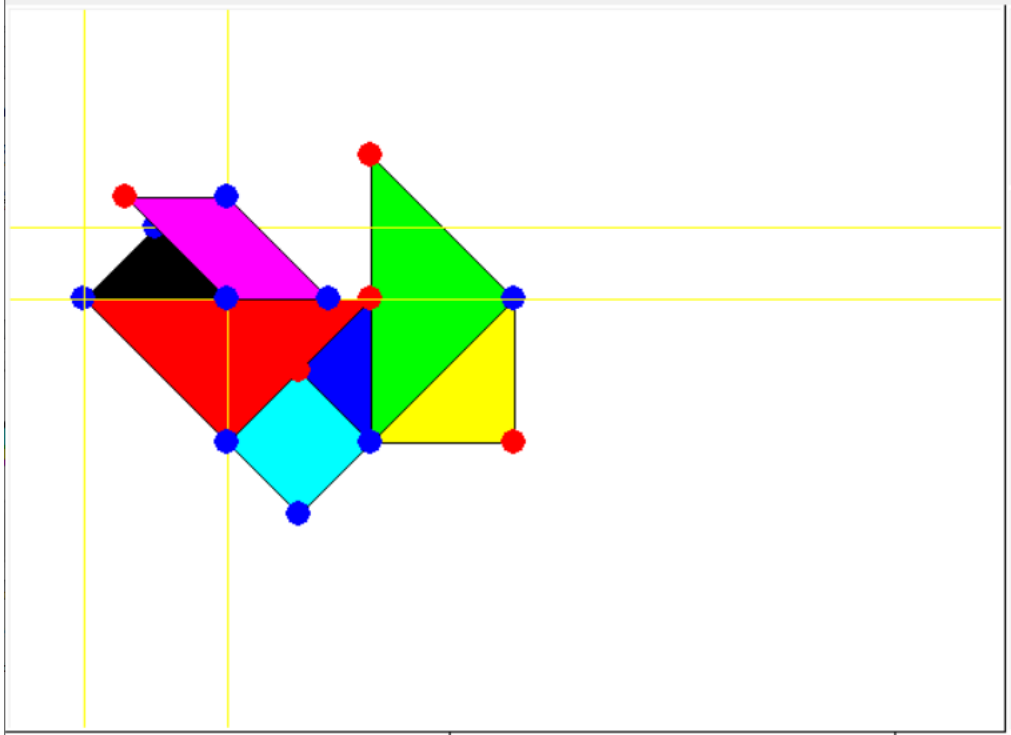




$\frac{2\pi r}{\sqrt{\pi r^2}} = 2\sqrt{\pi} \sim 3.5$ $\frac{4r}{\sqrt{r^2}} = 4$ $\frac{8r}{\sqrt{3r^2}} \sim 4.6$ $\frac{20}{\sqrt{14}} \sim 5.3$

Tangram Frame

Quit | Load | Save | Stop | Refresh | New | Problem | Solve!



Construction

All

Show Thinking

All Solutions

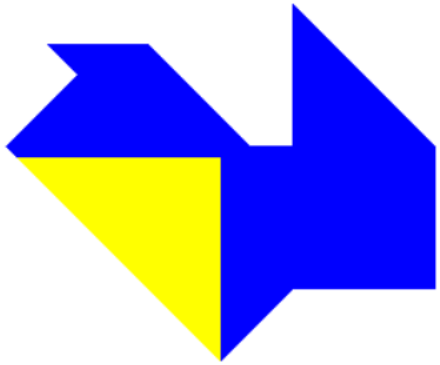
Available Tile Sets

Available Problems

Known Solutions

Tangram Frame

Quit | Load | Save | Stop | Refresh | New | Problem | Solve!



Construction

All

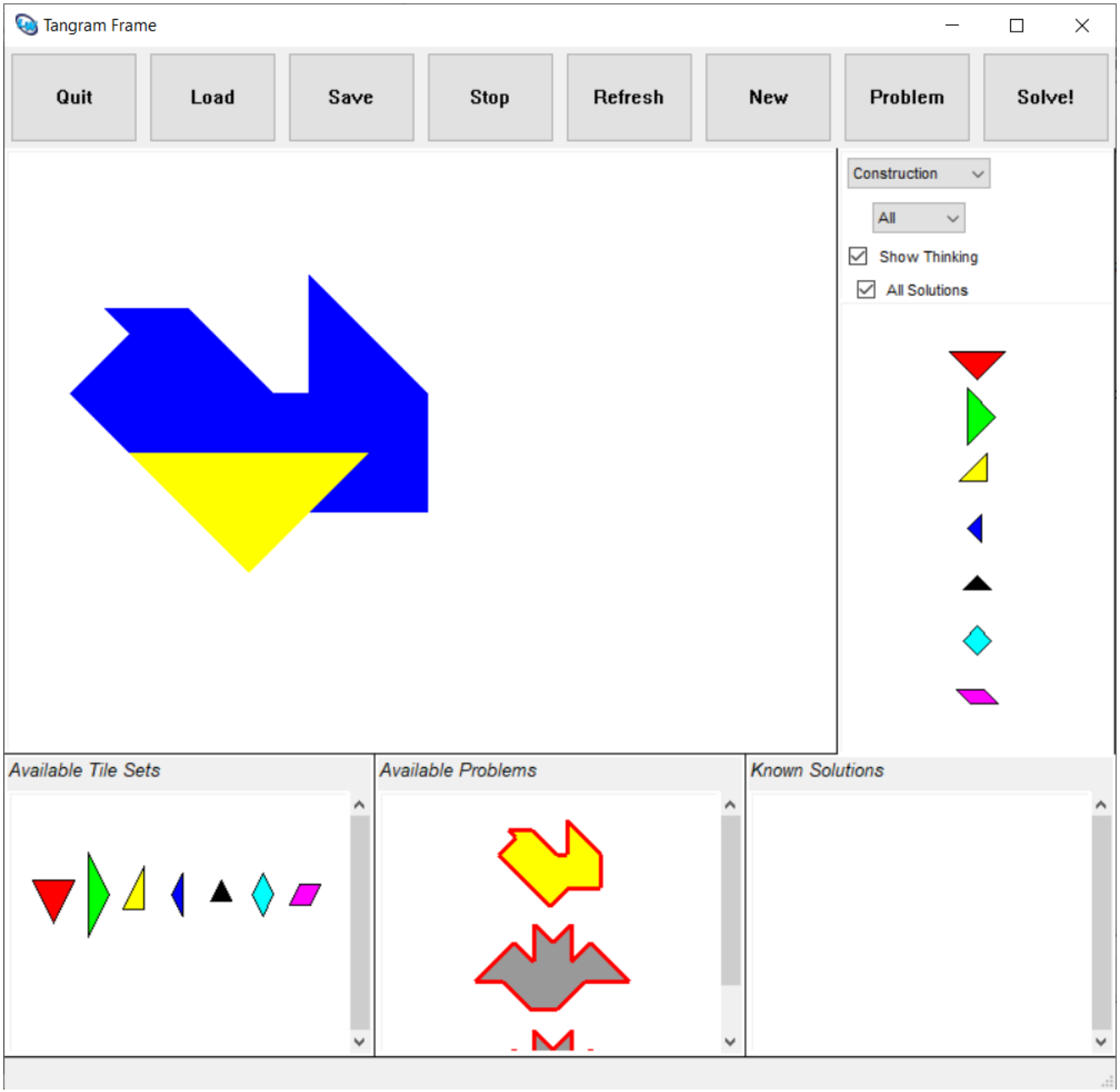
Show Thinking

All Solutions

Available Tile Sets

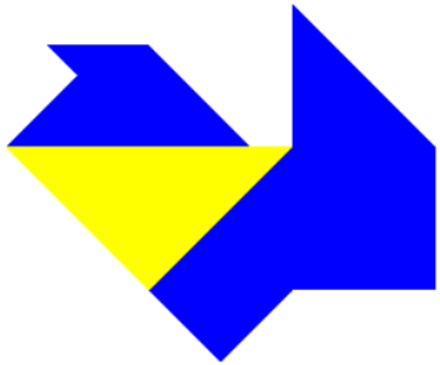
Available Problems

Known Solutions


Tangram Frame
Quit
Load
Save
Stop
Refresh
New
Problem
Solve!
Construction
All
Show Thinking
All Solutions
Available Tile Sets
Available Problems
Known Solutions

Tangram Frame

Quit | Load | Save | Stop | Refresh | New | Problem | Solve!



Construction

All

Show Thinking

All Solutions

Available Tile Sets

Available Problems

Known Solutions

Geometric Heuristics Continued

```
(let ((sorted (append (sort perfect-alignments #'tuple-> :key #'caar)
                      (subseq (sort alignments #'tuple-> :key #'caar)
                              ;; best n only ?
                              0 (length alignments)
                              ))))
```

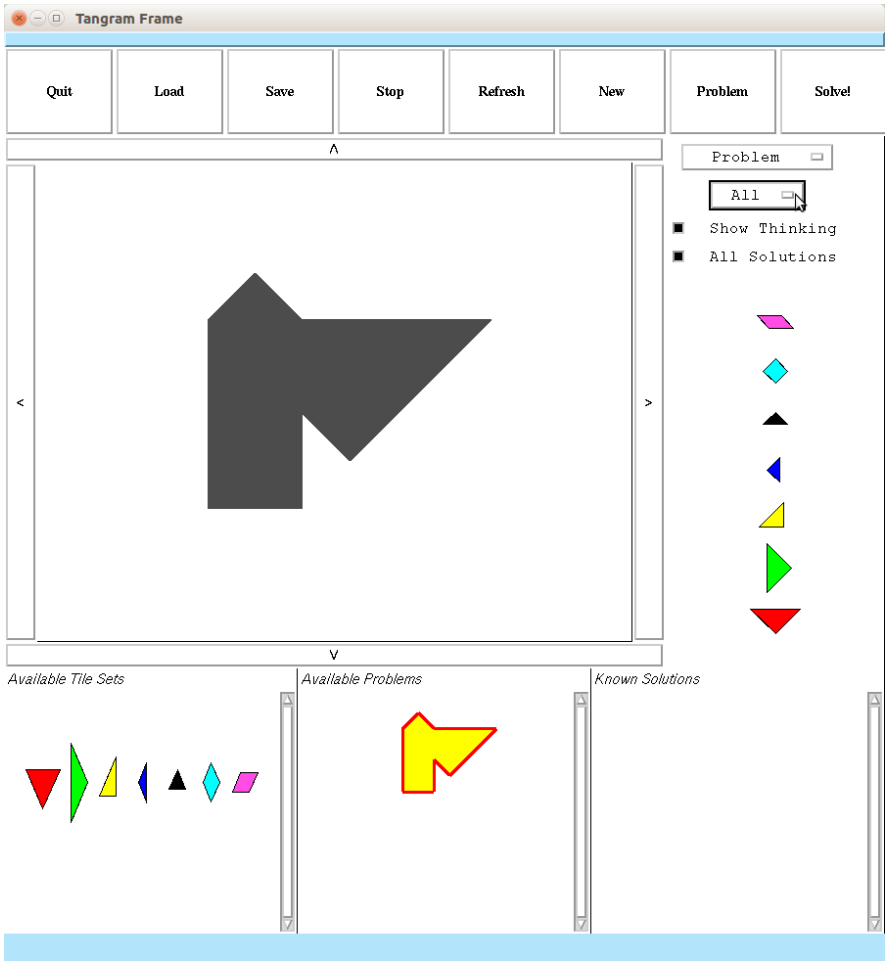
- How to compare two successor configurations wrt. different criteria?
  - Comparing **Apples with Pears**
  - Combine into one score
    - Weighted sum
    - Difficult, not normalized
  - Use **ordered list of criteria instead lexicographic ordering (see tuple->)**
  - **Score = (tile size, -compactness, - #interior vertices, …)**

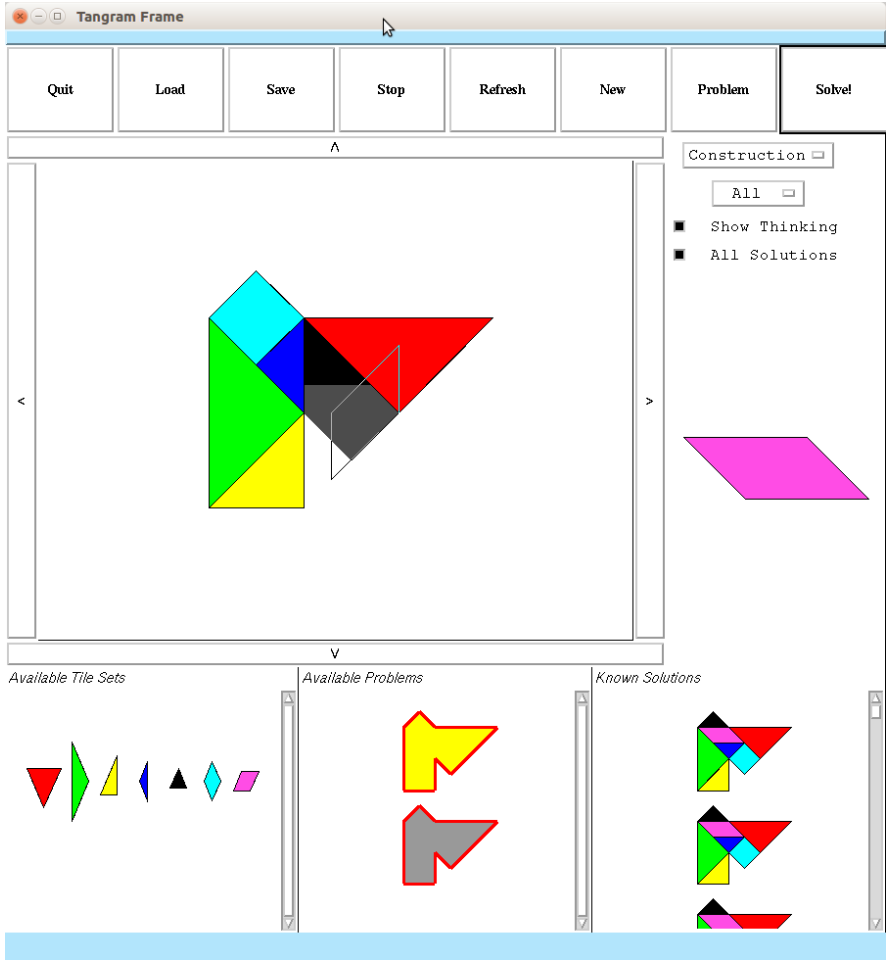
```
(defun tuple-> (a b)
  (when (and a b)
    (cond ((> (first a) (first b))
           t)
          ((= (first a) (first b))
           (tuple-> (rest a) (rest b)))
          (t nil))))
```

Scoring Function

```
(defun alignment-result-heuristic-for-polygon (aligned-polygon conf orig-polygon rem-tile-types)
  (cond ((symbolp conf) ; :match!

         (list (tangram-geometry:calculate-area aligned-polygon) 0))

        ((some #'(lambda (poly)
                   (every #'(lambda (tile-type)
                              (let ((tile (first tile-type))) ; use first reflected variant
                                (< (+ (tangram-geometry:calculate-area poly) 2)
                                   ;; account for inaccuracies...
                                   (tangram-geometry:calculate-area (shape tile)))))
                          rem-tile-types))
               conf)
         :bad )

        (t

         ;; score list for lexicographic tuple->-p sorting

         (list
          (tangram-geometry:calculate-area aligned-polygon)
          (- (count-if #'(lambda (p)
                           (inside-p p orig-polygon))
                       (point-list aligned-polygon)))
          (apply #'min (mapcar #'(lambda (x) (- (compactness x))) conf))
          (- (reduce #'+ (mapcar #'length (mapcar #'segments conf))))))))
```

Demo Video Heuristics





Further Essential Tricks

- Remove congruent successor configurations
  - Congruent configurations (the same configuration being generated from different input) occur frequently
  - Tiles have rotational symmetries
  - There are congruent tile types
  - Don‘t explore the same search space regions repeatedly
  - Reduce branching factor / search space
- Remove configurations that have polygons which are smaller than the smallest (remaining) tile
  - Backtrack early
  - Avoid trashing

Performance & Heuristics

Problem	Optimizations A & B	Optimizations A & ! B	Optimizations ! A & ! B
Standard Problem Solution Found at Branching Factor First 3 Levels	~ 3 mins **2705** **104, 76, 44**	~ 6 mins **4230** **104, 76, 44**	~ 27 mins **22788** **148, 112, 65**
Bat Problem	~ 1 min	~ 2.5 mins	**~ 45 mins (!)**
Other Test Problem	~ 10 seconds		

- **Without heuristics, none of these problems can be solved**
- ~15 Configurations / Second
- These **additional heuristics** are hightly effective as well
  Optimization A Remove congruent successor configurations
  Optimization B Remove successor configuration with polygons that are too small for any tile to cover / fit in

About Lisp – 1 / 3

- CLOS is used extensively
  - Geometric classes
  - Compositional hierachy
    - Segments from Points, Polygons from Segments, …
  - Dynamically affected by transformation matrix
  - Reader methods for `x`, `y`

```
(defmethod x ((point geom-point))
  (with-slots (x y affected-by-matrix-p) point
    (let ((val
           (if (and *matrix* affected-by-matrix-p)
               (with-slots (a b tx) *matrix*
                 (+ (* a x) (* b y) tx))
             x)))
      (my-round val))))

(defmethod y ((point geom-point))
  (with-slots (x y affected-by-matrix-p) point
    (let ((val
           (if (and *matrix* affected-by-matrix-p)
               (with-slots (c d ty) *matrix*
                 (+ (* c x) (* d y) ty))
             y)))
      (my-round val))))
```

=> all geometric functions are automatically aware of active transformations (e.g., `calculate-rcc-relation` next slide)

About Lisp – 2 / 3

- Transformations similar to CLIM

```
(setf (affected-by-matrix-p a) nil)
(setf (affected-by-matrix-p b) t)

(with-translation ( (- xf) (- yf) )
  (with-rotation ( phi )
    (with-translation ( xt yt )

      (when (and frame (show-thinking-p frame))
        (show-polygons (list b)
                       :clear-p nil
                       :inks (list +flipping-ink+)
                       :filled-p nil))

      (let* ((rel (calculate-rcc-relation a b)))
```

About Lisp – 3 / 3

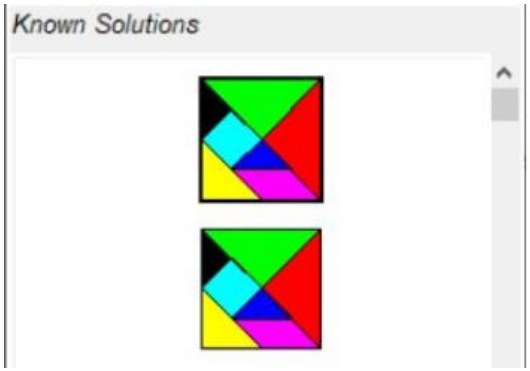
- Multi-Methods are nice for **Case Analysis**

```
(defmethod touches-p ((line geom-line) (poly geom-polygon))
  (and (intersects-p line poly)
       (or (lies-on-p line poly)
           (not (one-part-is-inside-p line poly)))))

(defmethod touches-p ((poly geom-polygon) (line geom-line))
  (touches-p line poly))

(defmethod touches-p ((chain1 geom-chain) (chain2 geom-chain))
  ;;; (error "TOUCHES-P ~A ~A: Not well-defined!" chain1 chain2))
  )

(defmethod-memo touches-p ((obj geom-chain) (poly geom-polygon))
    ((obj poly (trafo-id *matrix*)))
  (and (intersects-p obj poly)
       (every #'(lambda (segment)
                  (=> (intersects-p segment poly)
                      (touches-p segment poly)))
              (segments obj))))

(defmethod touches-p ((poly geom-polygon) (obj geom-chain))
  (touches-p obj poly))

(defmethod-memo touches-p ((poly1 geom-polygon) (poly2 geom-polygon))
    ((poly1 poly2 (trafo-id *matrix*)))
  (and (not (eq poly1 poly2))
       (intersects-p poly1 poly2)
       (every #'(lambda (segment)
                  (=> (intersects-p segment poly2)
                      (touches-p segment poly2)))
              (segments poly1))))
```

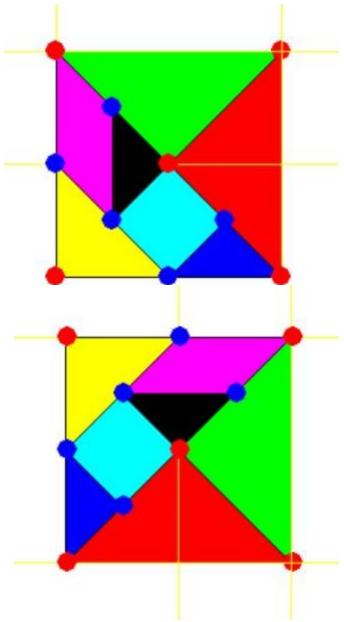
About CLIM

- CLIM is great for building prototypes quickly
- CLOS classes are presentations
  - Active objects that are alive on the screen (highlighting)
  - Automatically generated context menus etc.
  - You get A LOT of functionallity with minimal effort

Observations



- Frequently finds interesting solutions
- Is it **superhuman**? I guess it depends:
  - Search frequently is, not only since DNNs (e.g., Checkers, Chess / DeepBlue, …)
  - Compared to my tangram puzzle abilities, yes!
  - Not sure how fast people are
- Are other programs faster?
  - There are few available online… and they appeared only recently
  - Most of them are special-purpose and Tangram-specific
  - Whereas ours is generic and can (in principle) solve arbirtary tiling problems
  - Our program is complete (probably not the case for some of the others)
  - Sometimes, completeness has to be sacrificied for performance
- **Show Thinking** checkbox is useful for checking the progress
  - However, 60 % slower
  - The heuristics were devised by watching the search program and giving the program some „common sense“ and human insight

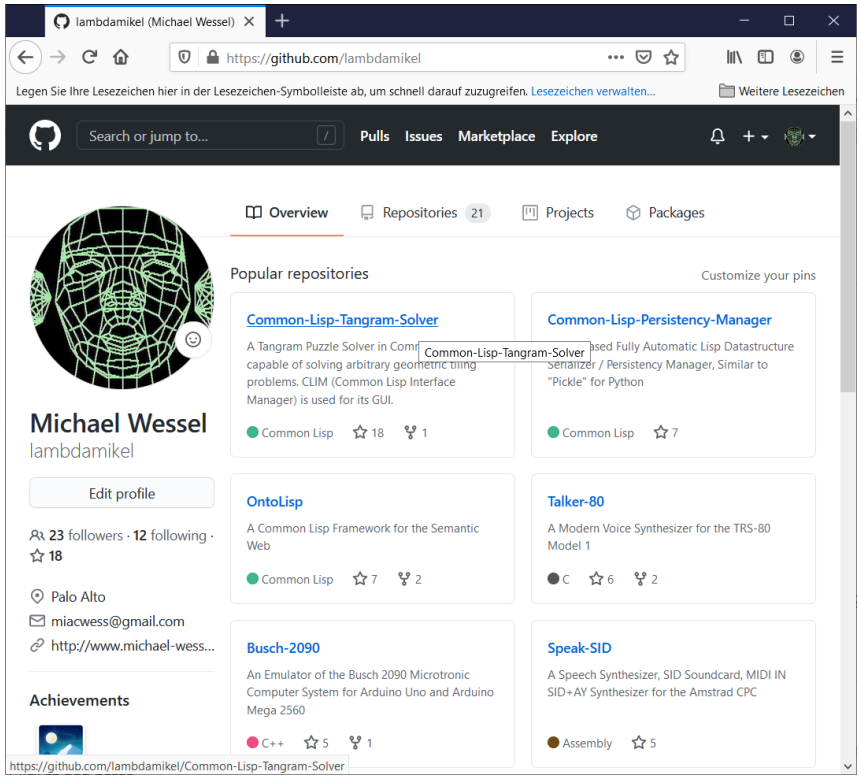


Wrap Up & Future Work

- **Q: Would I still be able to maintain and extent my Lisp code from almost 20 years ago?**
- **A: YES**
  - Lisp was fun to work with again after many years of absence
  - Lisp has good code maintainability
  - Concise, precise, semi-formal, small computational units (functions & methods)
  - The program is running much better than in 2003; I feel that, for now, **I finished what was started**
- The search space is very large
  - Getting the initial positions of the first two tiles right is essential
  - Else backtracking to them won‘t happen for a long time
  - Without heuristics no solutions can be found within 24 hours
- Future Work
  - SBCL / McCLIM, Penrose Tilings, …
  - More optimizations (Random Restarts, …)

Some Resources – Parens for You!

- Github repositories
  `https://github.com/lambdamikel`
  - Geometric substrate
  - Persistence module
  - Some reusable functions there for you
- Sourcecode and executables for Tangram
  - Try tweaking it, change heuristics! It‘s easy and inutitive
- Uses Lispworks (6.1) + CLIM
  - Linux, Windows
  - M1 Silicon Mac – Lispworks 7.1
  - **Thanks to Rainer Joswig (Lispm)!**



"A rollercoaster adventure. . . . A nail-biting narrative [of] the thrill of scientific discovery." —*NATURE*

THE SECOND KIND OF IMPOSSIBLE

THE EXTRAORDINARY QUEST FOR A NEW FORM OF MATTER

THE ROYAL SOCIETY • INSIGHT INVESTMENT
SCIENCE BOOK PRIZE 2019
SHORTLISTED

PAUL J. STEINHARDT

Thank You!

Questions?