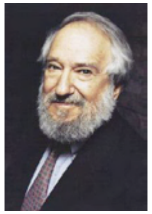
3D Turtle Geometry, Turtle Programs, Symmetry, Miter Joints

Computer Science Colloquium
10 December 2009, Eindhoven

Tom Verhoeff
Eindhoven University of Technology
Department of Mathematics & CS
`www.win.tue.nl/~wstomv`

"3D Turtle Geometry with an Application to Mitre and Fold Joints"
Accepted for *International Journal of Arts and Technology*

Seymour Papert (1960s): Turtle Graphics and Logo



- **Turtle Graphics**, mechanical, virtual
- **Logo** programming language: `Repeat 5 [ Forward 100 Left 72 ]`
- Goal: Enable children to do/enjoy computer programming
- **Turtle Geometry**: mathematical theory of turtle figures

2D Turtle Geometry

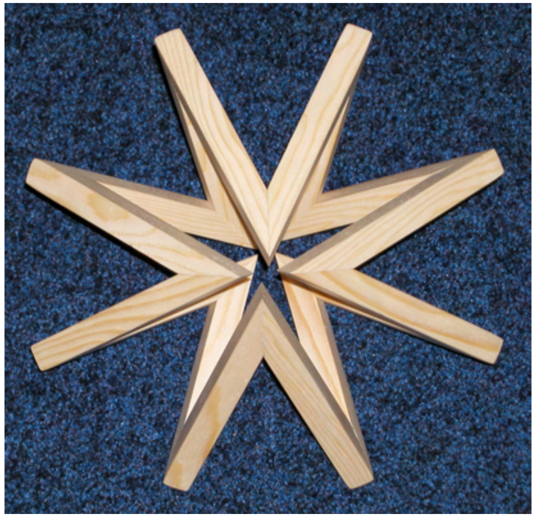
The book

> Harold Abelson and Andrea A. diSessa
> *Turtle geometry*:
> *The computer as a medium for exploring mathematics.*
> MIT Press, 1981.

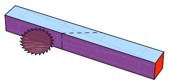
presents an exhaustive treatment of 2D Turtle Geometry,

including Turtle Graphics on curved surfaces.

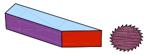
Mathematical Art by Koos Verhoeff



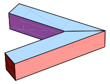
Miter Joints



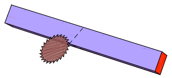
intact beam



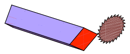
beveled at 30°



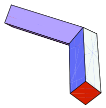
60° miter joint



intact beam
(rolled 45°)



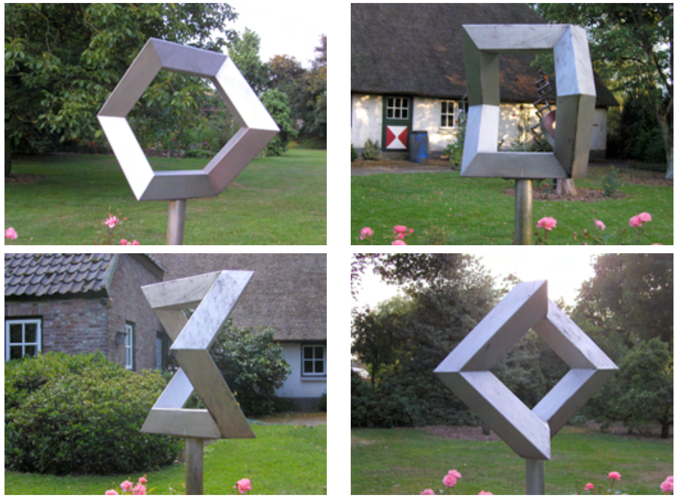
beveled at 60°



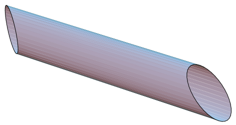
120° miter joint

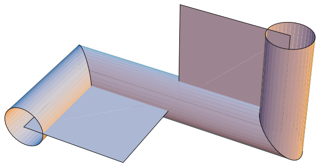
Mathematica Demonstrations Project: *Miter Joint and Fold Joint*

'Kliekje' (Eng.: 'Left over')



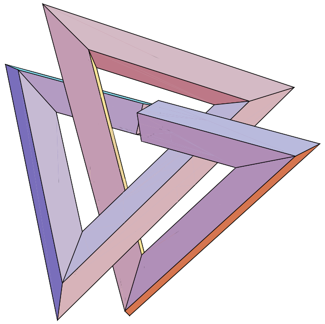
Spatial Mitering



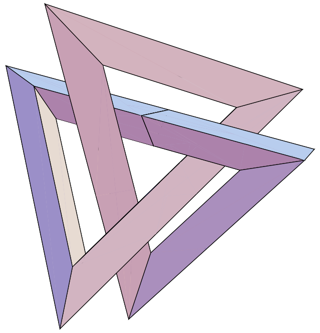


- **Corner plane** = plane spanned by adjacent segments
- **Torsion angle** = dihedral angle between adjacent corner planes

Closing the 3D Path



Square Cross Section



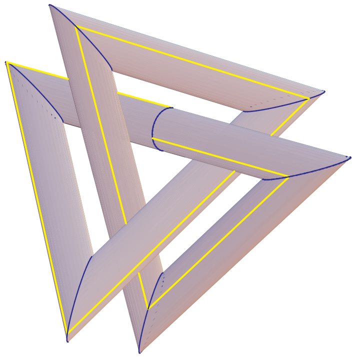
Equitriangular Cross Section

Mathematica Demonstrations Project: *Mitering a Closed 3D Path*

Miter Joint Torsion Invariance Theorem

Total amount of torsion is inherent property of polygonal path and does *not* depend on

- choice of initial segment
- initial rotation of cross section about center line
- shape of cross section

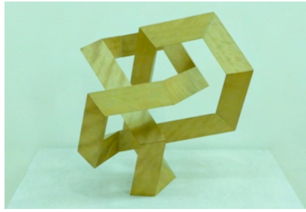


Mitering matches $\iff$ total torsion is symmetry of cross section

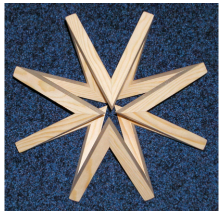
Three Techniques to Tackle Torsion

'Tinkering'

Lattice walking



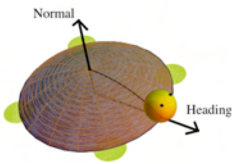
Constant torsion



3D Turtle Graphics

State of turtle:

- **Position** in space
- **Attitude** = (heading vector, normal vector)
- *Initial state*: in origin, heading to x^+, normal to z^+



Commands to change turtle state:

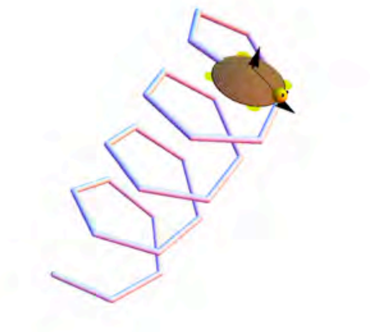
- $\mathit{Move}(d)$: move distance d in direction of heading (leaves **trace**)
- $\mathit{Turn}(\varphi)$: turn clockwise by angle φ about normal
- $\mathit{Roll}(\psi)$: roll clockwise by angle ψ about heading

Mathematica Demonstrations Project: *3D Flying Pipe-laying Turtle*

Spatial Figures

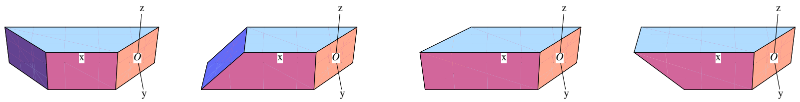
In 3D: roll angles provide additional freedom

All turn and roll angles equal yields a **helix**, which never closes



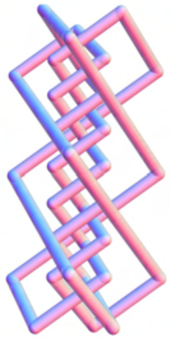
Some Macro Commands

$$
\begin{aligned}
\mathit{Segment}(d,\psi,\varphi) &:= \mathit{Move}(d)\,;\,\mathit{Roll}(\psi)\,;\,\mathit{Turn}(\varphi)\\
T_d &:= \mathit{Segment}(d,\ \ 0\ ,90^\circ)\\
R_d &:= \mathit{Segment}(d,\ 90^\circ,90^\circ)\\
P_d &:= \mathit{Segment}(d,180^\circ,90^\circ)\\
L_d &:= \mathit{Segment}(d,270^\circ,90^\circ)
\end{aligned}
$$



T_d trapezoid R_d right-handed P_d parallelogram L_d left-handed

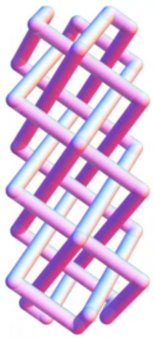
Artwork Described by Turtle Graphics: Spiralosaurus



$$\left(\ T_4^2\ ;\ L_9^2\ ;\ T_4^2\ ;\ R_3^6\ \right)^3$$

36 segments, 6 symmetries

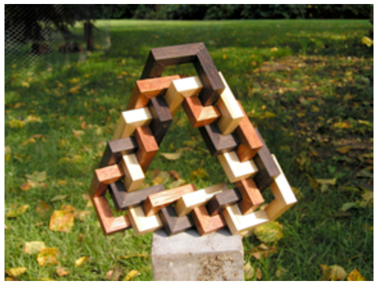
Artwork Described by Turtle Graphics: Braidwork

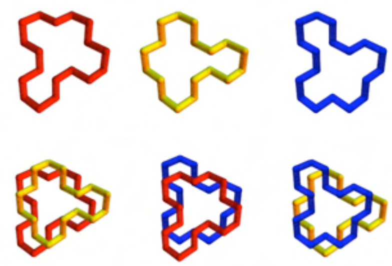


$$\left(\ L_1\ ; R_5\ ; R_6^2\ ; L_3\ ; R_1\ ; L_5\ ; L_6^2\ ; R_3\ \right)^3$$

30 segments, 6 symmetries (incl. mirror/upside-down)

Artwork Described by Turtle Graphics: Borromean Polylink

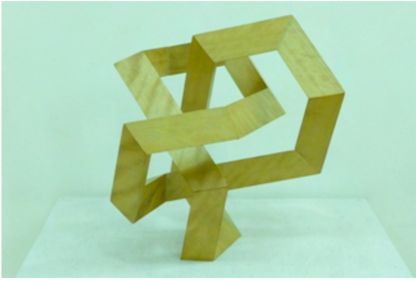




$R_3\,;\,P_3\,;\,L_3\,;\,P_3\,;\,R_3\,;\,L_3\,;\,P_3^2\,;\,R_3\,;\,P_3^2\,;\,L_3\,;\,R_3\,;\,P_3\,;\,L_3\,;\,P_3\,;\,R_3\,;\,L_3$

18 segments, 2 symmetries (per link)

Artwork Described by Turtle Graphics: Figure-Eight Knot



equitriangular cross section, 16 segments, 4 symmetries

$$R'_d \;:=\; \mathit{Segment}(d,\; 60°, \varphi) \qquad \varphi = \arctan(2\sqrt{2}) \approx 70.5°$$
$$L'_d \;:=\; \mathit{Segment}(d, -60°, \varphi)$$

$$\left(R'_2 \,;\, L'_1 \,;\, R'_1 \,;\, L'_1 \,;\, L'_2 \,;\, R'_1 \,;\, L'_1 \,;\, R'_1 \right)^2$$

Fundamentals of 3D Turtle Geometry: Program Semantics

Motion μ_p of program p is mapping $\mathbb{R} \to \mathbb{R}^3$ from time to space: $\mu_p(t)$ is position at time t (move at *unit speed*, instant turn/roll)

N.B. Mapping time to state will not work when turn/roll are instantaneous

Total duration δ_p of p's motion: $\delta_p = \sum_{\mathit{Move}(d)\,\in\, p} |d|$

Final attitude α_p of p is a tuple of heading and normal

Trace τ_p of p: $\tau_p = \{\, \mu_p(t) \mid 0 \le t \le \delta_p \,\}$ (set of points visited)

Empty program $\mathcal{I}$: no commands, leaving turtle in initial state, *identity* of sequential composition

Fundamentals of 3D Turtle Geometry: Program Properties

Closed program: $\mu_p(\delta_p) = \mu_p(0)$; (final *position* = initial *position*)
Open is not closed

Properly closed: closed and $\alpha_p = \alpha_{\mathcal{I}}$ (final *state* = initial *state*)
Closed program can be made *properly* closed by appending roll/turn/roll commands

Simple program: $\mu_p(t_1) \neq \mu_p(t_2)$ for $0 \le t_1 < t_2 < \delta_p$ (μ_p injective)

Being (properly) closed is a local property
Being simple is a global property

Trivial program: $|\tau_p| = 1$, i.e. $\tau_p = \{\, (0, 0, 0) \,\}$ (closed, simple)

Empty program $\mathcal{I}$ is trivial and properly closed

Simple Programs and Motion/Trace Relationship

Unique Motion Theorem for *simple open* programs:

$$\frac{p, q : \text{simple} \;\wedge\; \tau_p = \tau_q \;\wedge\; p : \text{open}}{\mu_p = \mu_q \;\wedge\; q : \text{open}}$$

Reverse motion $\tilde{\mu}_p$ of p: $\tilde{\mu}_p(t) = \mu_p(\delta_p - t)$

Two-Motion Theorem for *simple closed* programs:

$$\frac{p, q : \text{simple} \;\wedge\; \tau_p = \tau_q \;\wedge\; p : \text{closed}}{(\mu_p = \mu_q \;\vee\; \tilde{\mu}_p(t) = \mu_q(t)) \;\wedge\; q : \text{closed}}$$

Equivalence Relations for Turtle Programs

Motion equivalent: $p \overset{\mu}{\equiv} q \iff \mu_p = \mu_q$ (abstracts from program)

Final-attitude equivalent: $p \overset{\alpha}{\equiv} q \iff \alpha_p = \alpha_q$

Equivalent: $p \equiv q \iff p \overset{\mu}{\equiv} q \wedge p \overset{\alpha}{\equiv} q$ (for sequential composition)

Trace equivalent: $p \overset{\tau}{\equiv} q \iff \tau_p = \tau_q$ (abstracts from time)

Unique Motion Theorem rephrased:
For simple open programs, trace equivalence is motion equivalence

(Trace) congruent: $p \overset{c}{\equiv} q \iff \tau_p, \tau_q$ are congruent (via isometry; abstracts from placement of figure in space)

$$p \equiv q \Rightarrow p \overset{\mu}{\equiv} q \wedge p \overset{\alpha}{\equiv} q \qquad p \overset{\mu}{\equiv} q \Rightarrow p \overset{\tau}{\equiv} q \qquad p \overset{\tau}{\equiv} q \Rightarrow p \overset{c}{\equiv} q$$

Simplified Notation

When pressed for space, we abbreviate:

$$\begin{aligned} \mathcal{M}(d) &:= \mathit{Move}(d) \\ \mathcal{T}(\varphi) &:= \mathit{Turn}(\varphi) \\ \mathcal{R}(\psi) &:= \mathit{Roll}(\psi) \\ \mathcal{S}(d,\psi,\varphi) &:= \mathit{Segment}(d,\psi,\varphi) \\ \pi &= 180^\circ \end{aligned}$$

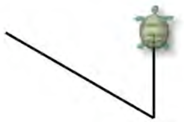
Omit ; for composition:

$$\mathcal{S}(d,\psi,\varphi) = \mathcal{M}(d)\ \mathcal{R}(\psi)\ \mathcal{T}(\varphi)$$

$$\mathcal{H} = \mathcal{T}(\pi)\ \mathcal{R}(\pi)$$

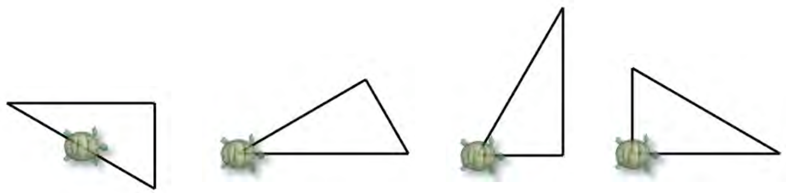
Example for Equivalences of Simple Open Programs

$$\begin{aligned} & \mathcal{T}(-30^\circ)\ \mathcal{M}(1)\ \mathcal{M}(1)\ \mathcal{T}(90^\circ)\ \mathcal{T}(30^\circ)\ \mathcal{M}(1)\ \mathcal{T}(90^\circ) \\ \equiv \quad & \{ \text{ put in standard form } \} \\ & \mathcal{R}(\pi)\ \mathcal{T}(30^\circ)\ \mathcal{S}(2,\pi,120^\circ)\ \mathcal{S}(1,0,90^\circ)\ \mathcal{R}(0) \\ \overset{\mu}{\equiv} \quad & \{ \text{ put in } \mu\text{-standard form } \} \\ & \mathcal{R}(\pi)\ \mathcal{T}(30^\circ)\ \mathcal{S}(2,\pi,120^\circ)\ \mathcal{S}(1,0,0) \\ \overset{c}{\equiv} \quad & \{ \text{ put in } c\text{-standard form } \} \\ & \mathcal{S}(2,0,120^\circ)\ \mathcal{S}(1,0,0) \\ \overset{c}{\equiv} \quad & \{ \text{ apply } \mathit{rev} \text{ and put in } c\text{-standard form } \} \\ & \mathcal{S}(1,0,120^\circ)\ \mathcal{S}(2,0,0) \end{aligned}$$

Example for Equivalences of Simple Properly Closed Programs

$$\begin{aligned} & \mathcal{T}(-30^\circ)\ \mathcal{S}(1,0,120^\circ)\ \mathcal{S}(1,0,90^\circ)\ \mathcal{S}(\sqrt{3},0,150^\circ)\ \mathcal{S}(1,0,30^\circ) \\ \overset{c}{\equiv} \quad & \{ \text{ put in } cc\text{-standard form } \} \\ & \mathcal{S}(2,0,120^\circ)\ \mathcal{S}(1,0,90^\circ)\ \mathcal{S}(\sqrt{3},0,150^\circ) \\ \overset{c}{\equiv} \quad & \{ \mathit{shift} \text{ once } \} \\ & \mathcal{S}(1,0,90^\circ)\ \mathcal{S}(\sqrt{3},0,150^\circ)\ \mathcal{S}(2,0,120^\circ) \\ \overset{c}{\equiv} \quad & \{ \mathit{shift} \text{ once more } \} \\ & \mathcal{S}(\sqrt{3},0,150^\circ)\ \mathcal{S}(2,0,120^\circ)\ \mathcal{S}(1,0,90^\circ) \end{aligned}$$



Properties of Equivalences and Sequential Composition

$$
\begin{array}{lcl}
p \overset{\alpha}{\equiv} p' \wedge q \overset{\alpha}{\equiv} q' & \Rightarrow & p\,;q \overset{\alpha}{\equiv} p'\,;q' \\
p \equiv p' \wedge q \overset{\mu}{\equiv} q' & \Rightarrow & p\,;q \overset{\mu}{\equiv} p'\,;q' \\
p \equiv p' \wedge q \equiv q' & \Rightarrow & p\,;q \equiv p'\,;q'
\end{array}
$$

$$\mathit{Move}(d) \overset{\alpha}{\equiv} \mathcal{I}$$

$$
\begin{array}{rcl}
\mathcal{I}\,;p & \equiv & p \\
p\,;\mathcal{I} & \equiv & p
\end{array}
$$

$$
\begin{array}{rcl}
p\,;\mathit{Turn}(\varphi) & \overset{\mu}{\equiv} & p \\
p\,;\mathit{Roll}(\psi) & \overset{\mu}{\equiv} & p
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{Turn}(\varphi)\,;p & \overset{c}{\equiv} & p \\
\mathit{Roll}(\psi)\,;p & \overset{c}{\equiv} & p
\end{array}
$$

Equivalence Properties of Basic Commands: 1, 2 (of 8)

1. *Turn* and *Roll* are periodic with period 360°:

$$
\begin{array}{rcl}
\mathit{Turn}(\varphi_1) \equiv \mathit{Turn}(\varphi_2) & \Longleftrightarrow & \varphi_1 = \varphi_2 \pmod{360^\circ} \\
\mathit{Roll}(\psi_1) \equiv \mathit{Roll}(\psi_2) & \Longleftrightarrow & \psi_1 = \psi_2 \pmod{360^\circ}
\end{array}
$$

2. Equivalence to the empty program $\mathcal{I}$:

$$
\begin{array}{rcl}
\mathit{Move}(d) \equiv \mathcal{I} & \Longleftrightarrow & d = 0 \\
\mathit{Turn}(\varphi) \equiv \mathcal{I} & \Longleftrightarrow & \varphi = 0 \pmod{360^\circ} \\
\mathit{Roll}(\psi) \equiv \mathcal{I} & \Longleftrightarrow & \psi = 0 \pmod{360^\circ}
\end{array}
$$

Equivalence Properties of Basic Commands: 3, 4 (of 8)

3. Adjacent commands of the same type can be merged:

$$
\begin{array}{rcll}
\mathit{Move}(d_1)\,;\mathit{Move}(d_2) & \equiv & \mathit{Move}(d_1 + d_2) & \text{provided } d_1 d_2 \geq 0 \\
\mathit{Turn}(\varphi_1)\,;\mathit{Turn}(\varphi_2) & \equiv & \mathit{Turn}(\varphi_1 + \varphi_2) & \\
\mathit{Roll}(\psi_1)\,;\mathit{Roll}(\psi_2) & \equiv & \mathit{Roll}(\psi_1 + \psi_2) &
\end{array}
$$

   Corollary: $\mathit{Turn}(180^\circ)$ and $\mathit{Roll}(180^\circ)$ are their own inverse.

4. Adjacent commands of the same type commute:

$$
\begin{array}{rcll}
\mathit{Move}(d_1)\,;\mathit{Move}(d_2) & \equiv & \mathit{Move}(d_2)\,;\,\mathit{Move}(d_1) & \text{provided } d_1 d_2 \geq 0 \\
\mathit{Turn}(\varphi_1)\,;\mathit{Turn}(\varphi_2) & \equiv & \mathit{Turn}(\varphi_2)\,;\,\mathit{Turn}(\varphi_1) & \\
\mathit{Roll}(\psi_1)\,;\mathit{Roll}(\psi_2) & \equiv & \mathit{Roll}(\psi_2)\,;\,\mathit{Roll}(\psi_1) &
\end{array}
$$

Equivalence Properties of Basic Commands: 5 (of 8)

5. When turtle rolls upside down, its turning sense looks reflected:

$$
\begin{array}{rcl}
\mathit{Roll}(180^\circ)\,;\mathit{Turn}(\varphi) & \equiv & \mathit{Turn}(-\varphi)\,;\mathit{Roll}(180^\circ) \\
\mathit{Turn}(\varphi) & \equiv & \mathit{Roll}(180^\circ)\,;\mathit{Turn}(-\varphi)\,;\mathit{Roll}(180^\circ)
\end{array}
$$

   Similarly for half-turn and roll sense:

$$
\begin{array}{rcl}
\mathit{Turn}(180^\circ)\,;\mathit{Roll}(\psi) & \equiv & \mathit{Roll}(-\psi)\,;\mathit{Turn}(180^\circ) \\
\mathit{Roll}(\psi) & \equiv & \mathit{Turn}(180^\circ)\,;\mathit{Roll}(-\psi)\,;\mathit{Turn}(180^\circ)
\end{array}
$$

   And also for half-turn and move sense:

$$
\begin{array}{rcl}
\mathit{Turn}(180^\circ)\,;\mathit{Move}(d) & \equiv & \mathit{Move}(-d)\,;\mathit{Turn}(180^\circ) \\
\mathit{Move}(d) & \equiv & \mathit{Turn}(180^\circ)\,;\mathit{Move}(-d)\,;\,\mathit{Turn}(180^\circ)
\end{array}
$$

Equivalence Properties of Basic Commands: 6, 7 (of 8)

6. Adjacent *Move* and *Roll* commands commute:

$$\mathit{Move}(d)\ ;\ \mathit{Roll}(\psi) \quad\equiv\quad \mathit{Roll}(\psi)\ ;\ \mathit{Move}(d)$$

7. *Turn*–*Move* and *Turn*–*Roll* do not commute, unless one of them is equivalent to $\mathcal{I}$, or in the special case

$$\mathit{Turn}(180^\circ)\ ;\ \mathit{Roll}(180^\circ) \quad\equiv\quad \mathit{Roll}(180^\circ)\ ;\ \mathit{Turn}(180^\circ)$$

Corollary: $\mathcal{H} := \mathit{Turn}(180^\circ)\ ;\ \mathit{Roll}(180^\circ)$ is its own inverse.
$\mathcal{H}$ (half-loop) is equivalent to $\mathit{Dive}(180^\circ)$.

Equivalence Properties of Basic Commands: 8 (of 8)

8. Every trivial program (without $\mathit{Move}(d)$ for $d \neq 0$) is equivalent to program of the form

$$\mathit{Roll}(\psi)\ ;\ \mathit{Turn}(\varphi)\ ;\ \mathit{Roll}(\psi')$$

with $0 \le \varphi \le 180^\circ$ and $-180^\circ < \psi, \psi' \le 180^\circ$.

Angles ψ, φ, ψ' are uniquely determined, when requiring

$$\varphi = 0 \pmod{180^\circ} \;\Rightarrow\; \psi = 0$$

Corollary: There exists a rule to rewrite

$$\mathit{Roll}(\psi_1)\ ;\ \mathit{Turn}(\varphi_1)\ ;\ \mathit{Roll}(\psi_2)\ ;\ \mathit{Turn}(\varphi_2)$$

into the form above, involving 'messy' (inverse) trigonometry

Some Algebraic Calculations

How $\mathcal{H}$ behaves in combination with the basic commands:

$\mathcal{H}\ \mathcal{M}(d)$	$\mathcal{H}\ \mathcal{T}(\varphi)$	$\mathcal{H}\ \mathcal{R}(\psi)$
$\equiv$	$\equiv$	$\equiv$
$\mathcal{T}(\pi)\ \mathcal{R}(\pi)\ \mathcal{M}(d)$	$\mathcal{T}(\pi)\ \mathcal{R}(\pi)\ \mathcal{T}(\varphi)$	$\mathcal{T}(\pi)\ \mathcal{R}(\pi)\ \mathcal{R}(\psi)$
$\equiv$	$\equiv$	$\equiv$
$\mathcal{T}(\pi)\ \mathcal{M}(d)\ \mathcal{R}(\pi)$	$\mathcal{T}(\pi)\ \mathcal{T}(-\varphi)\ \mathcal{R}(\pi)$	$\mathcal{T}(\pi)\ \mathcal{R}(\psi)\ \mathcal{R}(\pi)$
$\equiv$	$\equiv$	$\equiv$
$\mathcal{M}(-d)\ \mathcal{T}(\pi)\ \mathcal{R}(\pi)$	$\mathcal{T}(-\varphi)\ \mathcal{T}(\pi)\ \mathcal{R}(\pi)$	$\mathcal{R}(-\psi)\ \mathcal{T}(\pi)\ \mathcal{R}(\pi)$
$\equiv$	$\equiv$	$\equiv$
$\mathcal{M}(-d)\ \mathcal{H}$	$\mathcal{T}(-\varphi)\ \mathcal{H}$	$\mathcal{R}(-\psi)\ \mathcal{H}$

Standardisation Theorem

Every program p is *equivalent* to exactly one program in **standard form**:

$$\sigma(p) \quad=\quad \mathcal{R}(\psi_0)\ \mathcal{T}(\varphi_0)\ \mathcal{S}(d_1,\psi_1,\varphi_1)\ \ldots\ \mathcal{S}(d_n,\psi_n,\varphi_n)\ \mathcal{R}(\psi_{n+1})$$

where $n \ge 0$ and the parameters satisfy these constraints:

C1: $d_i > 0$ for $1 \le i \le n$,

C2: $-\pi < \psi_i \le \pi$ for $0 \le i \le n+1$,

C3: $0 \le \varphi_i \le \pi$ for $1 \le i \le n$,

C4: $\varphi_i \neq 0$ for $1 \le i < n$, i.e., between $\mathcal{M}$ commands, and

C5: if $\varphi_i = 0 \pmod{\pi}$ then $\psi_i = 0$ for $0 \le i \le n$.

Furthermore, if p is *simple*, then

C4a: $\varphi_i \neq \pi$ for $1 \le i < n$. (**strict standard form**)

Proof of Standardization Theorem

Existence: by induction on the program's structure

Base cases: $\mathcal{I}$, $\mathcal{M}(d)$, $\mathcal{T}(\varphi)$, $\mathcal{R}(\psi)$

Inductive step: $p\ q$, where neither is empty, massage $\sigma(p)\ \sigma(q)$

Only relies on basic properties (so these are *complete*)

Unicity: consider first difference between two standard forms

Simple case: if $\varphi_i = \pi$, then $\sigma(p)$ not simple

- In general, it is *messy* to determine $\sigma(p)$ (cf. Basic Property 8)
- It is *easy* to check whether p is in standard form

Reduced Standard Form

By dropping *trailing* $\mathcal{R}, \mathcal{T}$ commands, we infer that every program p is *motion equivalent* to exactly one program in **μ-standard form**

$$\sigma_\mu(p) \quad = \quad \mathcal{R}(\psi_0)\ \mathcal{T}(\varphi_0)\ \mathcal{S}(d_1, \psi_1, \varphi_1)\ \ldots\ \mathcal{S}(d_n, \psi_n, \varphi_n)$$

where d_i, ψ_i, and φ_i satisfy constraints C1–C5, and $\psi_n = \varphi_n = 0$.

By dropping *leading* $\mathcal{R}, \mathcal{T}$ commands as well, every program p is *congruent* to a program in **c-standard form**

$$\sigma_c(p) \quad = \quad \mathcal{S}(d_1, \psi_1, \varphi_1)\ \ldots\ \mathcal{S}(d_n, \psi_n, \varphi_n)$$

where d_i, ψ_i, and φ_i satisfy constraints C1–C5, and $\psi_1 = \psi_n = \varphi_n = 0$. This representation is *not* unique.

Determining Program Equivalence and Motion Equivalence

For equivalence, consider standard form (Standardization Theorem):

$$p \equiv q \iff \sigma(p) = \sigma(q)$$

For *motion* equivalence, consider μ-standard form:

$$p \overset{\mu}{\equiv} q \iff \sigma_\mu(p) = \sigma_\mu(q)$$

Determining Trace Equivalence

For *simple open* programs (Unique Motion Theorem): see μ-equivalence

Reversal $rev(p)$ of p: reverse the order of constituting $\mathcal{M}$, $\mathcal{T}$, and $\mathcal{R}$ commands *and* reverse signs of their parameters. Then

$$\delta_{rev(p)} \quad = \quad \delta_p$$

If p is *properly* closed, then:

$$\begin{aligned} \mu_{rev(p)} &= \tilde{\mu}_p \\ \mu_{rev(p)}(t) &= \mu_p(\delta_p - t) \end{aligned}$$

(motion of the reversal of p equals the reverse motion of p)

For *simple properly closed* programs (Two-Motion Theorem):

$$p \overset{\mathcal{T}}{\equiv} q \quad \iff \quad p \overset{\mu}{\equiv} q \ \vee\ rev(p) \overset{\mu}{\equiv} q$$

Reversal of c-Standard Form

$rev(\mathcal{R}(\pi)\ \mathcal{T}(30°)\ \mathcal{S}(2,\pi,120°)\ \mathcal{S}(1,0,90°)) \equiv \mathcal{R}(0)\ \mathcal{T}(90°)\ \mathcal{S}(1,\pi,120°)\ \mathcal{S}(2,0,150°)\ \mathcal{R}(\pi)$

$$\begin{aligned}\sigma_c(p) &= \mathcal{S}(d_1,\psi_1,\varphi_1)\ \ldots\ \mathcal{S}(d_n,\psi_n,\varphi_n)\\ &\ \text{where } \psi_1 = \psi_n = \varphi_n = 0\end{aligned}$$

$$\sigma_c(rev(p)) = \mathcal{S}(d'_1,\psi'_1,\varphi'_1)\ \ldots\ \mathcal{S}(d'_n,\psi'_n,\varphi'_n)$$

where for $1 \le i \le n$

$$\begin{aligned}d'_i &= d_{n+1-i}\\ \psi'_i &= \psi_{n+1-i}\\ \varphi'_i &= \varphi_{n-i} \qquad \text{N.B. } \varphi_0 = 0\end{aligned}$$

Congruence of Simple Open Programs

Only two kinds of mappings possible between their traces:

1. the initial positions are paired and so are the final positions, or
2. the initial position of one trace is paired with the final position of the other trace and conversely.

The second case is reduced to the first by comparing $rev(p)$ with q.

Translations are not relevant

Rotations are taken care of by comparing their *reduced* standard forms, since rotating p's trace is accomplished by an $\mathcal{R}\ \mathcal{T}\ \mathcal{R}$ prefix.

Reflection of c-Standard Form

Reflection $refl(p)$ of program p (in x, y-plane): negate all roll angles

c-standard form of reflection:

$$\begin{aligned}\sigma_c(p) &= \mathcal{S}(d_1,\psi_1,\varphi_1)\ \ldots\ \mathcal{S}(d_n,\psi_n,\varphi_n)\\ \sigma_c(refl(p)) &= \mathcal{S}(d_1,-\psi_1,\varphi_1)\ \ldots\ \mathcal{S}(d_n,-\psi_n,\varphi_n)\end{aligned}$$

where roll angle $-\pi$ is replaced by π.

Simple open programs p, q are *congruent* if and only if at least one of the following four equalities holds:

$$\begin{aligned}\sigma_c(p) &= \sigma_c(q)\\ \sigma_c(refl(p)) &= \sigma_c(q)\\ \sigma_c(rev(p)) &= \sigma_c(q)\\ \sigma_c(refl(rev(p))) &= \sigma_c(q)\end{aligned}$$

Congruence of Simple Properly Closed Programs

Cyclic Permutation Congruence Theorem (CPC):

> If program $p\ q$ (that is, p followed by q) is *properly* closed, then so is $q\ p$, and we have $p\ q \overset{c}{\equiv} q\ p$.

Simple *properly* closed p is *congruent* to a **cc-standard form**:

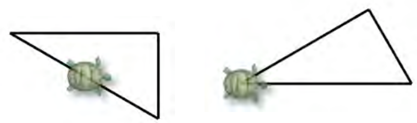
$$\sigma_{cc}(p) = \mathcal{S}(d_1,\psi_1,\varphi_1)\ \mathcal{S}(d_2,\psi_2,\varphi_2)\ \ldots\ \mathcal{S}(d_n,\psi_n,\varphi_n)$$

where the parameters satisfy these constraints ($1 \le i \le n$):

CC1: $d'_i > 0$,

CC2: $-\pi < \psi'_i \le \pi$,

CC3: $0 < \varphi'_i < \pi$.



This form is not uniquely determined.

Congruence of Simple Properly Closed Programs

Cyclic shift $\mathit{shift}(p)$ of *properly* closed p in cc-*standard form*:

$$\begin{array}{rcl} p & = & \mathcal{S}(d_1,\psi_1,\varphi_1)\ \mathcal{S}(d_2,\psi_2,\varphi_2)\ \ldots\ \mathcal{S}(d_n,\psi_n,\varphi_n) \\ \mathit{shift}(p) & = & \mathcal{S}(d_2,\psi_2,\varphi_2)\ \ldots\ \mathcal{S}(d_n,\psi_n,\varphi_n)\ \mathcal{S}(d_1,\psi_1,\varphi_1) \end{array}$$

Note: $\mathit{shift}(p)$ is then also in cc-standard form

Properly closed simple programs p, q in cc-*standard form* are congruent if and only if at least one of the following equalities holds:

$$\begin{array}{rcl} \mathit{shift}^k(p) & = & q \\ \mathit{refl}(\mathit{shift}^k(p)) & = & q \\ \mathit{rev}(\mathit{shift}^k(p)) & = & q \\ \mathit{refl}(\mathit{rev}(\mathit{shift}^k(p))) & = & q \end{array}$$

where k ranges from 0 to $n-1$ with n the number of segments in p.

If p and q differ in number of segments, then they are not congruent.

Summary of Equivalence Determination

Relation	Condition	Criterion
$p \equiv q$		$\sigma(p) = \sigma(q)$
$p \overset{\mu}{\equiv} q$		$\sigma_\mu(p) = \sigma_\mu(q)$
$p \overset{\tau}{\equiv} q$	simple, open	$p \overset{\mu}{\equiv} q$
$p \overset{\tau}{\equiv} q$	simple, properly closed	$p \overset{\mu}{\equiv} q \vee \mathit{rev}(p) \overset{\mu}{\equiv} q$
$p \overset{c}{\equiv} q$	simple, open	apply σ_c to p, $\mathit{rev}(p)$, $\mathit{refl}(p)$, $\mathit{refl}(\mathit{rev}(p))$, and q
$p \overset{c}{\equiv} q$	simple, properly closed	apply σ_{cc} to $\mathit{shift}^k(p)$, $\mathit{rev}(\mathit{shift}^k(p))$, $\mathit{refl}(\mathit{shift}^k(p))$, $\mathit{refl}(\mathit{rev}(\mathit{shift}^k(p)))$, and q

Symmetry

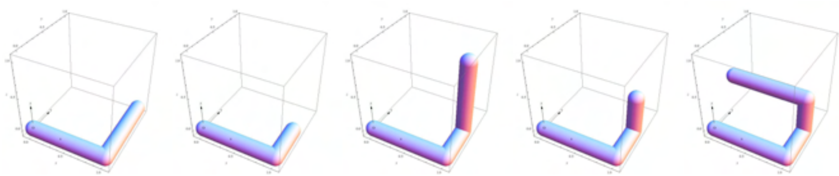
Symmetry is self-congruence

Test the program for congruence with itself

Every satisfied equality corresponds to a symmetry of the trace

- rev and shift^k correspond to rotation
- refl corresponds to reflection

Symmetries of Simple Open Programs



$\mathcal{S}(1,0,90°)$ $\mathcal{S}(1,0,0)$	$\mathcal{S}(1,0,90°)$ $\mathcal{S}(^1/_2,0,0)$	$\mathcal{S}(1,0,90°)$ $\mathcal{S}(^1/_2,90°,90°)$ $\mathcal{S}(1,0,0)$	$\mathcal{S}(1,0,90°)$ $\mathcal{S}(^1/_2,90°,90°)$ $\mathcal{S}(^1/_2,0,0)$	$\mathcal{S}(1,0,90°)$ $\mathcal{S}(^1/_2,90°,90°)$ $\mathcal{S}(^1/_2,-90°,90°)$ $\mathcal{S}(1,0,0)$

I	refl
rev	$\mathit{refl}\ \mathit{rev}$

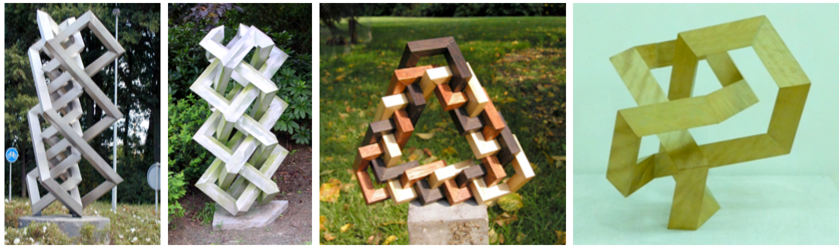
I	refl

I	
rev	

I	

I	
	$\mathit{refl}\ \mathit{rev}$

Symmetries of Simple Properly Closed Artwork



Object	Symmetries	#
Spiralosaurus	I, shift^{12}, shift^{24}, $\mathit{rev\ shift}^{6}$, $\mathit{rev\ shift}^{18}$, $\mathit{rev\ shift}^{30}$	6
Braidwork	I, shift^{10}, shift^{20}, $\mathit{refl\ shift}^{5}$, $\mathit{refl\ shift}^{15}$, $\mathit{refl\ shift}^{25}$	6
Borromean Polylink	I, $\mathit{rev\ shift}^{-1}$	2
Figure-Eight Knot	I, shift^{8}, $\mathit{refl\ shift}^{4}$, $\mathit{refl\ shift}^{12}$	4

Miterability Theorem for Closed 3D Polygons

For a simple properly closed program in standard closed form, the total amount of torsion to determine **closure of the mitering** equals the *sum of all roll angles, taking roll signs into account*.

Let the tip of the normal vector trace out an edge at angle φ_0:

$$
\begin{aligned}
& \mathcal{S}(d_1,\psi_1,\varphi_1)\ \mathcal{S}(d_2,\psi_2,\varphi_2)\ \ldots \\
\equiv\ & \mathcal{R}(\psi_0)\ \mathcal{R}(-\psi_0)\ \mathcal{S}(d_1,\psi_1,\varphi_1)\ \mathcal{S}(d_2,\psi_2,\varphi_2)\ \ldots \\
\equiv\ & \mathcal{R}(\psi_0)\ \mathcal{S}(d_1,-\psi_0+\psi_1,\varphi_1)\ \mathcal{S}(d_2,\psi_2,\varphi_2)\ \ldots \\
\equiv\ & \ldots\ \mathcal{S}(d_1,-\psi_0+\psi_1,\varphi_1)\ \mathcal{R}(\psi_0-\psi_1)\ \mathcal{R}(-\psi_0+\psi_1)\ \mathcal{S}(d_2,\psi_2,\varphi_2)\ \ldots \\
\equiv\ & \ldots\ \mathcal{S}(d_2,-\psi_0+\psi_1+\psi_2,\varphi_2)\ \ldots
\end{aligned}
$$

The turtle finishes off with

$$\mathcal{S}(d_n,-\psi_0+\Psi,\varphi_n)\ \mathcal{R}(\psi_0-\Psi)\ \mathcal{R}(-\psi_0+\Psi)$$

where $\Psi = \sum_{i=1}^{n}\psi_i$. Total torsion $= -\psi_0 + \Psi + \psi_0 = \Psi$ (CPC)

Foldability Theorem for Closed 3D Polygons

For a simple properly closed program in standard closed form, the total amount of torsion needed to determine **closure of the folding** equals the *alternating sum of all roll angles, taking roll signs into account*.

Let the tip of the normal vector trace out an edge at angle φ_0:

$$
\begin{aligned}
& \mathcal{S}(d_1,\psi_1,\varphi_1)\ \mathcal{S}(d_2,\psi_2,\varphi_2)\ \ldots \\
\equiv\ & \mathcal{R}(\psi_0)\ \mathcal{R}(-\psi_0)\ \mathcal{S}(d_1,\psi_1,\varphi_1)\ \mathcal{S}(d_2,\psi_2,\varphi_2)\ \ldots \\
\equiv\ & \mathcal{R}(\psi_0)\ \mathcal{S}(d_1,-\psi_0+\psi_1,\varphi_1)\ \mathcal{S}(d_2,\psi_2,\varphi_2)\ \ldots \\
\equiv\ & \ldots\ \mathcal{S}(d_1,-\psi_0+\psi_1,\varphi_1)\ \mathcal{R}(-\psi_0+\psi_1)\ \mathcal{R}(\psi_0-\psi_1)\ \mathcal{S}(d_2,\psi_2,\varphi_2)\ \ldots \\
\equiv\ & \ldots\ \mathcal{S}(d_2,\psi_0-\psi_1-\psi_2,\varphi_2)\ \ldots
\end{aligned}
$$

The turtle finishes off with

$$\mathcal{S}(d_n,(-1)^n(\psi_0-\Psi),\varphi_n)\ \mathcal{R}((-1)^n(\psi_0-\Psi))\ \mathcal{R}(-(-1)^n(\psi_0+\Psi))$$

where $\Psi = \sum_{i=1}^{n}(-1)^i\psi_i$.

Foldability Theorem for Closed 3D Polygons: Result

Total torsion $= -(-1)^n(\psi_0-\Psi)+\psi_0$

Even n: total torsion $= -(\psi_0-\Psi)+\psi_0 = \Psi$, independent of φ_0

Odd n: total torsion $= (\psi_0-\Psi)+\psi_0 = 2\psi_0-\Psi$, depends on ψ_0

Hence, for odd n, folding can always be made to close by appropriate choice of ψ_0

In fact, in *two ways*: with and without Möbius twist

Also see *Mitering a Closed 3D Path* (Mathematica Demonstrations Project)

Conclusion

- 3D variant of Turtle Graphics, including a roll command
- Some artwork elegantly described by turtle programs
- Various equivalences (program semantics)
- Algebraic reasoning about equivalence in 3D Turtle Geometry
- A standard form for each equivalence
- Correspondence between symmetries of figure and symmetries of program in standard form
- Proofs of the miter/fold joint torsion invariance theorems