The Unknot Game

Cameron Browne

Smart Games, Kontich, Belgium; cambolbro@gmail.com

Abstract

Unknot is a hyper-casual video game in which players are presented with convoluted drawings of the *unknot* (i.e. the trivial knot or plane circle) which they must undo with as few moves as possible. The game serves a serious purpose as a research tool for investigating the visual presentation of mathematical knots, by correlating player performance with knot drawing metrics to identify practical guidelines for the effective presentation of knots. This work involved devising a new approach for generating and smoothing knot diagrams at interactive speeds, and implementing two special knot moves aimed to be intuitive for players. Cognitive Load Theory (CLT) informs both the design of the game and the drawing of the knots, to minimise factors that might otherwise mask the effects being investigated.

Visualising Mathematical Knots

Knot theory has been an active field of research for over a century with more than a million related publications [13], most of which are illustrated. However, there has been surprisingly little investigation into *how* mathematical knots might be best visually presented to make them most comprehensible to the viewer.

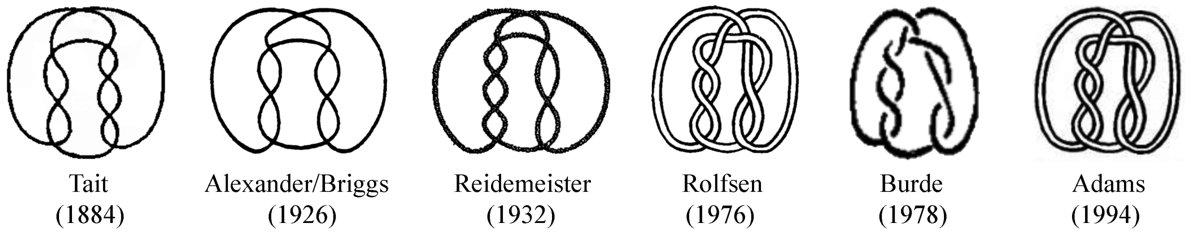

Figure 1: *Drawings of knot* 8_4 *from significant knot tables.*

Consider Figure 1, which shows drawings of the knot denoted 8_4 from some of the most historically significant knot tables throughout the literature. It is encouraging that this knot is consistently shown in the same basic form – which is not always the case! – but note that the 1926 drawing by Alexander and Briggs (from [1, p. 583]) contains an error and only has seven crossings. It is remarkable that such an error can occur for such a simple knot, even in fundamental work by pioneers of the field.

Figure 2 shows the topologically distinct *combinatorial embeddings* of this knot's underlying abstract graph, each obtained by choosing a particular *face* (i.e. region) as perimeter. Any of these embeddings may be chosen to draw the knot, which raises the question: *which embedding is most comprehensible for the viewer and therefore less likely to cause visual misinterpretations and misclassifications?*

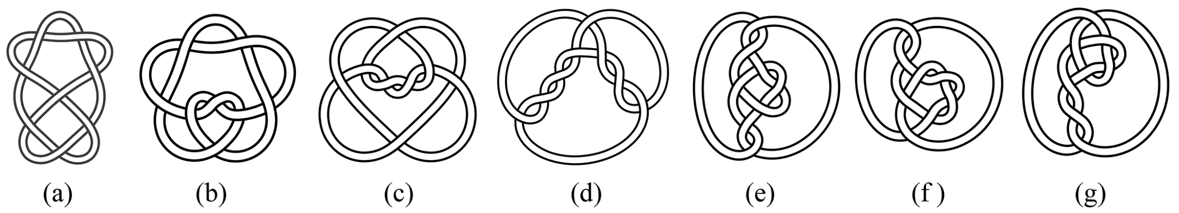

Figure 2: *Topologically distinct embeddings of knot* 8_4. *Which is visually simplest?*

Embedding (d) as shown in most knot tables highlights the knot's *twist regions* , which may be meaningful to knot theorists, but embedding (a) with its more regular layout may appear visually simpler to the lay observer. These embeddings are in fact listed in order of visual complexity according to three simple rules suggested in a previous Bridges paper [4], which are to maximise: 1) the number of perimeter crossings, 2) internal symmetry, and 3) perceived perimeter continuity. It is tempting to empirically test these ideas with a visual questionnaire, but such qualitative approaches often prove unreliable for visual domains [5].

Instead, consider the observation by De Toffoli and Giardino that knot diagrams are *dynamic* forms that invite the viewer to mentally apply the inherent moves [6]. It is a logical next step to present knots as interactive games that players can virtually manipulate, potentially providing insight into their understanding of the knots and allowing quantitative testing for effects of knot presentation on player performance.

The Unknot Game

Unknot is a hyper-casual video game in which players are presented with convoluted forms of the *unknot* (i.e. the trivial knot or plane circle) that they must undo to a circle with as few moves as possible. Figure 3 shows how challenges may be presented to the player (this is a "hard unknot" from Tuzun and Sikora [17]).

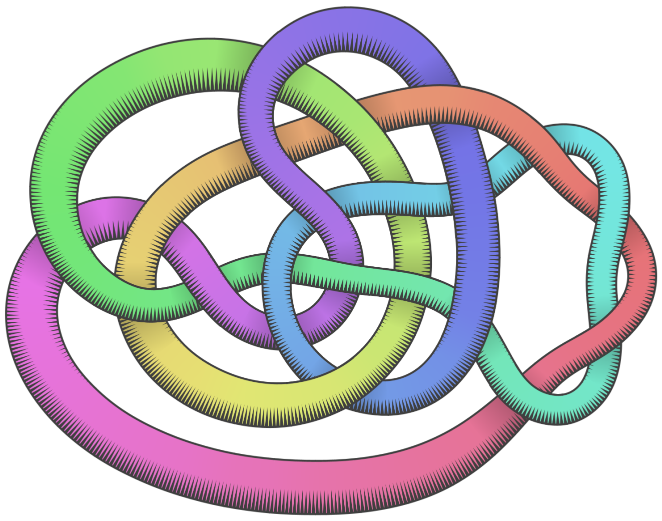

Figure 3: *Game view of the Tuzun-Sikora unknot.*

Each turn the player selects a point on the knot path, following which the move that most simplifies that local part of the knot is applied (if any) to remove as many crossings as possible, then the diagram is re-smoothed. If there is more than one such move, then the move that travels over the least area is made. This process is repeated until all crossings are removed and the knot is reduced to a circle in the plane.

The game is designed to make moves easy and engaging; each move can resolve large parts of the knot in surprising ways and produce a hypnotic effect as the diagram re-smooths itself, providing micro-rewards for the player. Most challenges can be solved with a few carefully chosen moves and games are quick.

Cognitive Load Theory

In attempting to gauge the player's understanding of a knot from their performance in the game, it is useful to consider the *cognitive load* (CL) experienced with each challenge. Cognitive load theory (CLT) examines the use of working memory when performing cognitive tasks [15] and distinguishes three main types of CL:

1. *Intrinsic:* Cognitive load due to the inherent complexity of the task.
2. *Extraneous:* Cognitive load due to the way the task is presented, especially nonessential components.
3. *Germane:* Learning schema (strategies) about the task to help reduce intrinsic cognitive load.

CLT informs the design of the game and its *user interface* (UI) to minimise CL experienced by the player and strip away superfluous elements that might otherwise mask the effects being investigated. The game has a minimal interface and intuitive moves so the player can fully concentrate on understanding the knot in front of them. CLT also informs the drawing styles in which knots are rendered, and highlights the competing objectives of visual simplicity (to minimise extraneous CL) and visual richness (to enhance the experience and allow the formation of germane strategies). These points are elaborated in the following sections.

Slip and Shrink Moves

Figure 4 shows the famous Reidemeister moves from classic knot theory [12], which are *atomic* moves that safely transform one knot diagram into another without modifying the underlying knot type. While these moves allow any convoluted diagram of the unknot to be reduced to a plane circle, their application can be non-intuitive, especially for *hard unknots* that require a complicating move before they can be simplified [10].

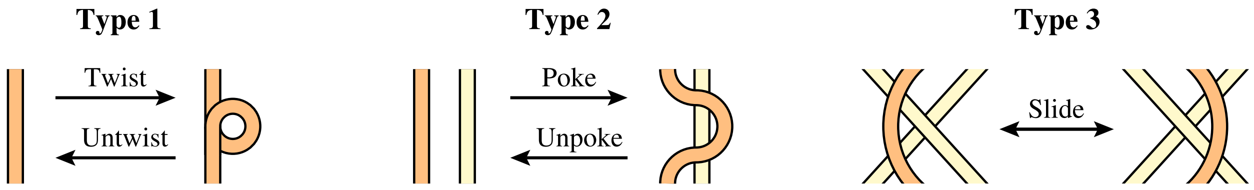

Figure 4: *The three types of Reidemeister move.*

Instead, the *Unknot* game uses two special moves dubbed "Slip" and "Shrink" that condense multiple Reidemeister moves into more convenient *compound* moves intended to be more intuitive for players. These respect the notion of *element interactivity* from CLT, which dictates that the number of component elements in a cognitive task and the way they interact affect the intrinsic cognitive load [16].

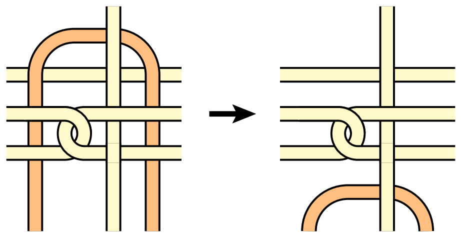

Figure 5: *A simplifying Slip move.*

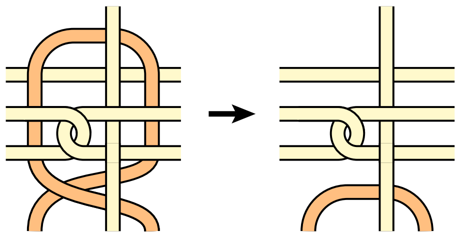

Figure 6: *A Shrink move (always simplifies).*

The Slip move (Figure 5) involves slipping a consecutive *run* of arcs past itself or any number of other arcs that can be cleanly separated into distinct "over" and/or "under" layers, such that no arcs straddle both layers (the Poke, Unpoke and Slide are special cases). The Shrink move (Figure 6) is similar to a Slip move except that the moving run shrinks to a point where it crosses itself (the Untwist is a special case).

Similar moves have already been proposed; the Shrink move is equivalent to the Z_1 move described in [11] and the Slip move is essentially a merger of the Z_2 and Z_3 moves but in which only a single run – selected by the player – actually moves. These compound moves reduce the number of trivial actions required to simplify the diagram, allowing the player to focus their mental effort on deciding *which* moves to make.

The Z moves[1] described in [11] efficiently resolve all diagrams of the unknot to which they've been applied, even hard unknots, with monotonically non-complicating moves. For example, the hard unknot shown in Figure 7, called the Monster, requires ten Reidmeister moves to resolve – the first of which adds two crossings [14] – but can be resolved with only three Z moves (or two Slips and a Shrink) as shown. Note that only simplifying Slip and Shrink moves are implemented in the game, to focus the task of the player, so some unknot diagrams will not be resolvable with the available moves; such cases are pre-tested and discarded.

[1] Together with accompanying moves denoted C and $\tilde{C}$ to handle special cases.

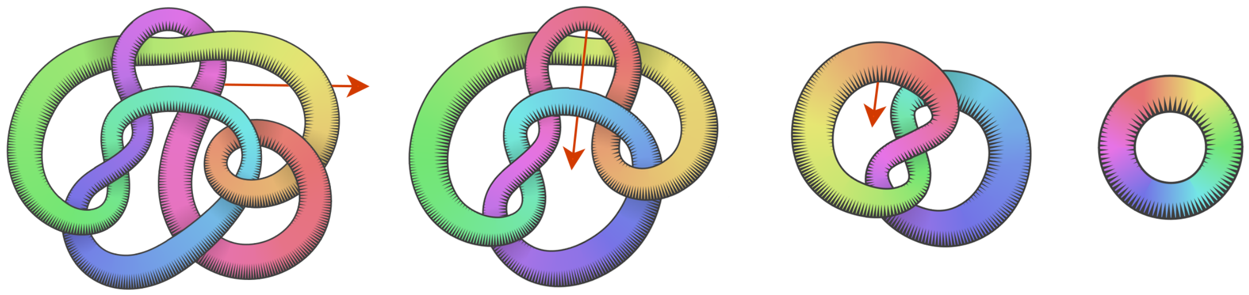
Figure 7: *Resolving the Monster with two Slips and a Shrink.*

Triangular Grid Diagrams

Each knot is represented internally as a *signed Gauss code* which lists its crossings in order of traversal with underpasses negated. From the Gauss code, it is straightforward to obtain the knot's underlying 4-valent *abstract graph*, in which each edge is an arc between consecutive crossings, using an algorithm described by Kauffman [9] and due to Dehn [7]. Choosing a face in this graph as perimeter constitutes a combinatorial embedding (as shown in Figure 2) but its geometric embedding in the plane is yet to be determined.

A *grid diagram* is obtained for each knot using a Monte Carlo Tree Search (MCTS) approach [3] in which the faces of the abstract graph are iteratively added to a triangular grid, from the largest deepest face outwards, using semi-random simulations. This *forward model* grows diagrams without the need for expensive geometric error checks and corrections, and the triangular grid offers a good balance between growth freedom and curve fairness, making the process fast and robust. Figure 8 (a) shows a triangular grid diagram obtained for the example shown above. The algorithm will be fully described in a separate article.

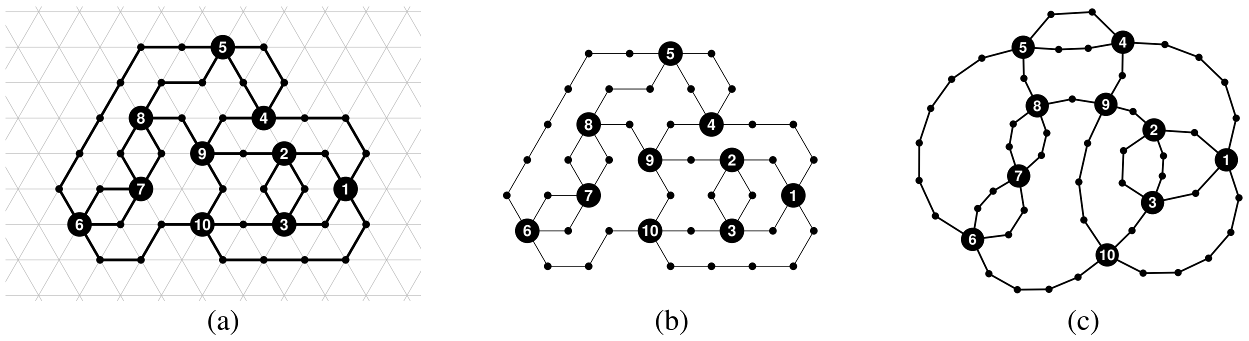

Figure 8: *Triangular grid diagram, control polygon and smoothed control polygon.*

The grid diagram has 4-valent vertices at the knot's crossings and 2-valent vertices along the arcs between crossings, and is converted directly into a *control polygon* that describes the knot path (b) with coincident vertices at the crossing points where it self-intersects. This control polygon is then smoothed as the player watches to give the drawing's final relaxed shape (c) as described below. Importantly, the triangular grid diagram layout closely matches the final relaxed shape, which minimises the amount of smoothing required.

Control Polygon Smoothing

The smoothing step relaxes the knot's control polygon into an aesthetically pleasing shape that minimises curvature and maximises the even spread of arcs across the drawing area. Smoothing is performed over a number of iterations in which the location of each vertex V_j is adjusted to a more desirable position according to the following properties, until the total displacement falls below a certain threshold:

1. *Step Length:* Each vertex V_j tries to make the lengths of its two incident edges match. This is approximated by adding to V_j a displacement equal to the displacement between its projection P_j onto the *short diagonal* $\overline{V_{j\pm1}}$ between its previous and next vertices and the midpoint M_{ik} of this short diagonal (Figure 9). This matches incident edge lengths without collapsing each V_j inwards.
2. *Curvature:* Each vertex V_j tries to make its curvature match the average curvature of its previous and next vertices. This is approximated by adding to V_j a displacement that would move it to the point along its projection onto its short diagonal $\overline{V_{j\pm1}}$ that is the same distance from this line as the average of the distances of the previous and next vertices from their short diagonals $\overline{V_{i\pm1}}$ and $\overline{V_{k\pm1}}$ (Figure 10).
3. *Proximity:* Each vertex V_j is displaced away from the closest point on any line segment along the other arcs in its two incident faces, by an amount proportional to the inverse square of its distance (except for crossing vertices, which take the average displacement of their four adjacent neighbours). This encourages the polygon to spread evenly across the drawing area.
4. *Frame:* Each vertex V_j is displaced away from the closest edge of the view frame (horizontal or vertical) an amount proportional to the inverse of its distance. This encourages the polygon to fit the view frame.

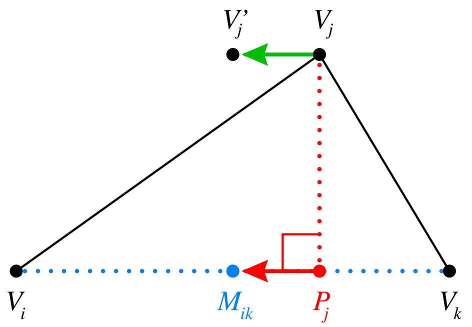

Figure 9: *Smoothing by step length.*

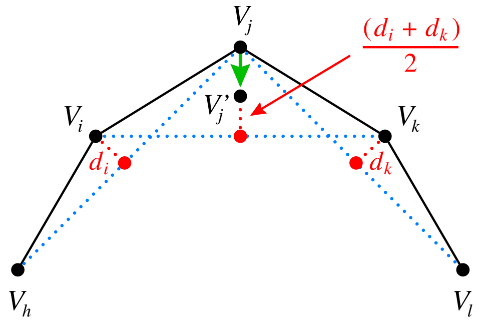

Figure 10: *Smoothing by curvature.*

The displacements for each vertex V_j are accumulated over all calculations, averaged with their previous and next vertices, then clamped at 45% of the distance to the nearest other point before being applied. This minimises the danger of displacements creating accidental self-intersections in the control polygon, to the extent that it has proven unnecessary to even check for these during the smoothing process. The two coincident vertices at each crossing are both displaced by the average of their amounts to keep them coincident.

After all displacements have been applied, the updated polygon is corrected for drift and spin, then at regular (but staggered) intervals the number of vertices in each arc is checked and adjusted if needed. For each arc A_n, a vertex is inserted or removed if its number of vertices $|A_n|$ differs from the expected number given by its geometric length $||A_n||$ divided by the polygon's average edge length L_{avg}. This smoothing process is approximate but produces good results and converges quickly in practice.

Curve Generation

The knot is finally drawn, as shown in Figure 11, using cubic B-Spline curves offset from the main control polygon (a). For each vertex V_j, an offset distance o_j is calculated based on its proximity to the nearest other point, and left and right offset points are then generated for V_j perpendicular to its short diagonal, to create left (green) and right (red) offset control polygons (b). The two cubic B-Spline curves defined by these offset control polygons describe the path borders of the final knot shape to be drawn (c). The curves are approximated by their control polygons rather than interpolated through them to avoid undue oscillations.

An *overpass region* is created for each crossing from the four curve sections that enclose it, inflated by a small amount to hide underlying visual artefacts. When the knot is drawn, each overpass obscured by another part of the path is redrawn after being clipped to its overpass region, to correctly draw all overpasses.

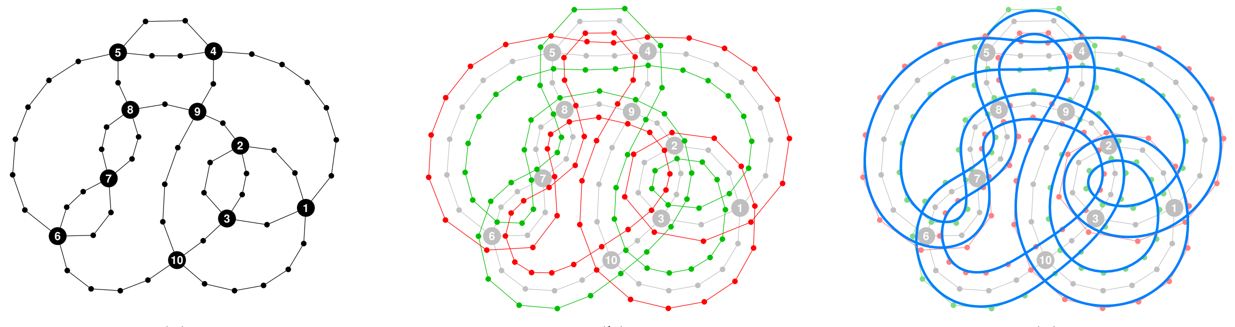

Figure 11: *Main control polygon, offset control polygons and final offset B-Splines.*

Rendering Styles

The *Unknot* game provides a number of rendering styles to not only cater for players' personal tastes but to explore the possible relationship between drawing style and player performance. These include the monochrome styles shown in Figure 12: flat fill with overpass borders excised, flat fill with dark borders, and hatched shading to enhance the 2.5D over/under effect.

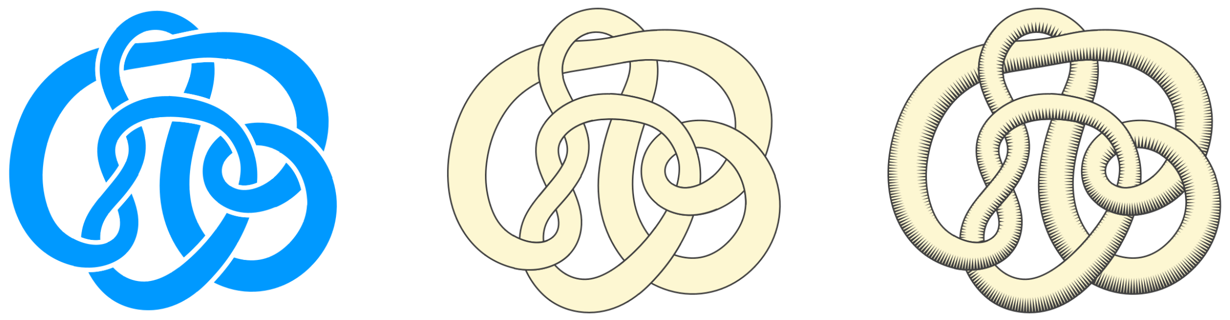

Figure 12: *Monochrome rendering styles: excised, bordered, and hatched.*

A variety of *rainbow scale* styles, which colour the knot path as a gradient through the hue component of the HSV colour space, are also provided (Figure 13). The rainbow scale is problematic for scientific visualisation [2] but works well as a visual cue to help distinguish sections of the knot path, thus facilitating new germane strategies at the cost of additional extraneous CL. The rainbow scale also allows the convenient identification of landmarks, e.g. the solution shown in Figure 7 could be described as "pink, red, orange".

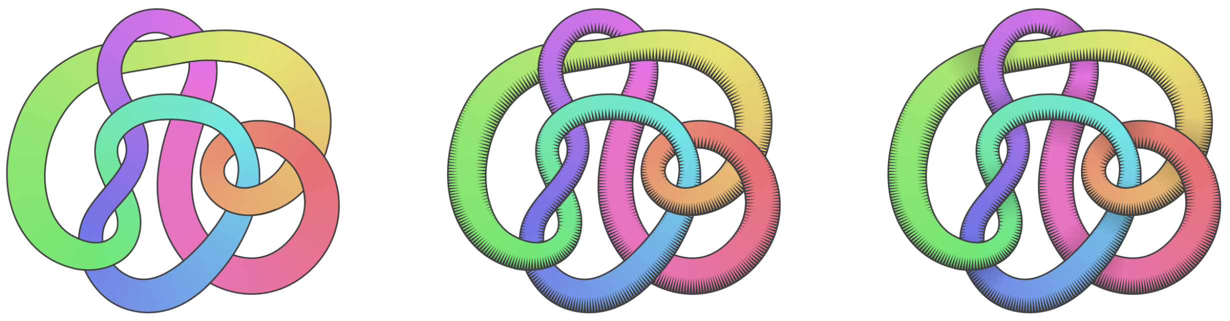

Figure 13: *Rainbow scale rendering styles: flat, hatched, and hatched with underpass darkening.*

Variable offsetting can be used to thicken the knot path based on local proximity to give a "fun" organic game-like look, which fills the view frame more evenly without cluttering the busier parts of the drawing. The player can turn this off to give a (thin) constant width path for a more classic look (as per Figure 2).

Challenge Generation

For each *challenge* in the game, a knot of appropriate difficulty is chosen at random from a pre-generated database, then perimeter face and writhe also chosen at random. Challenges are generated for a given number of crossings C and repetition count R by starting with the unknot's Gauss code (an empty string "") and applying semi-randomly chosen Reidemeister moves to it. Complicating moves are chosen with higher frequency while the code contains fewer than C crossings and simplifying moves are chosen with higher frequency while the code contains more than C crossings. This process continues until the R^{th} time that the code contains exactly C crossings (with Twists removed), after overshooting and compensating several times.

The program then tests whether each resulting knot can be unknotted using only simplifying Slip and Shrink moves and discards those that cannot. Generating challenges with Reidemeister moves but then testing them with simplifying Slip and Shrink moves increases the variety of challenges and reduces the occurrence of obvious solutions that simply reverse their generating sequence. Each challenge's *difficulty* is estimated by its number of crossings (inherent CL), minimum solution length (compressibility) and number of optimal moves compared to possible moves (tension). Each challenge's *quality* is estimated by its variation from a strict over/under weave (alternation) and the degree to which optimal moves are non-obvious (obfuscation).

Challenges are stored to file using the compact *alphabetic Dowker Thistlethwaite* (DT) format that assigns a letter to each crossing with case indicating sign [8], but extended with numeric superscripts that indicate multiples of 26 to add (to support more than 26 crossings). For example, the alphanumeric DT code for the Monster shown in Figure 7 is "BFIGHAJDCe".

Performance

The following timings indicate the running speed of a Java prototype of the *Unknot* game for mid-range knots with 25 crossings (challenges will range from 1 to 50 crossings). These approximate timings were taken on a standard consumer machine (Apple Mac Air with M3 chip and 16 GB RAM) performing drawing tasks in the foreground (UI) thread and non-trivial calculations in background (worker) threads:

- Grid diagram generation ~20 *ms*.
- Control polygon smoothing ~0.1 *ms* per iteration over ~1,200 iterations (~150 vertices).
- Move generation ~30 *ms*.
- Drawing to screen ~40 *ms* (1,600 x 1,200 pixels, rainbow scale with hatching and underpass darkening).

Most of the challenges tested loaded, smoothed and generated moves in under 1 second with an animation rate of ~20 *fps*, which is an interactive speed suitable for real-time play. Future speed improvements might be achieved by delegating smoothing calculations to the GPU to perform in parallel on a per-vertex basis.

Future Work

A core aim of this project is to investigate the following research questions:

RQ 1 *What effect does embedding choice have on player performance in unknotting?*

RQ 2 *What effect does rendering style have on player performance in unknotting?*

Anonymised play traces will be collected from players who agree to this, with each game played generating several data points. Indicators of player performance include the time taken per turn, success rate, number of undos/restarts/exits, *etc.* There will be no longitudinal study so data can be fully anonymised.

Player performance will be correlated with the variables involved – chosen embedding and writhe, diagram geometry, symmetry, rendering style, *etc.* – to address the research questions. Expected outcomes include guidelines for choosing the optimal embedding of knots and presenting them effectively. RQ 1 is the central research question but RQ 2 can also be investigated with the same data at little additional cost.

Summary and Conclusion

While originally intended as a meditative art piece, the *Unknot* game has taken on a serious purpose as a research tool for investigating the visual perception of mathematical knots. The game is intended to be engaging and entertaining, while at the same time collecting anonymised play traces from consenting players to explore the research questions. This will hopefully suggest guidelines for the effective presentation of knots in order to minimise their perceived visual complexity and reduce the potential for misinterpretations in what is already a visually confusing topic. This paper describes a complete approach for producing aesthetically pleasing knot drawings in real time, which could have broader artistic applications beyond the game described here. *Unknot* is planned for release in the second half of 2025 for web browsers and mobile devices.

Acknowledgements

Thanks to Rolf Vandoren from Smart for supporting this work. Thanks also to Frederic Maire, Stephen Tavener, Alain Brobecker, Néstor Romeral Andrés, Helen Gilbert and the anonymous reviewers for feedback.

References

[1] J. W. Alexander and G. B. Briggs. "On Types of Knotted Curves." *Annals of Mathematics*, vol. 28, no. 1/4, 1926–1927, pp. 562–586.

[2] D. Borland and R. M. Taylor. "Rainbow Color Map (Still) Considered Harmful." *IEEE Computer Graphics and Applications*, vol. 27, no. 2, 2007, pp. 14–17.

[3] C. Browne, *et al.* "A Survey of Monte Carlo Tree Search Methods." *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, 2012. pp. 1–43.

[4] C. Browne. "Nice Knots." *Bridges Proceedings*, Aalto, Finland, Aug. 1–5, 2022, pp. 309–312. https://archive.bridgesmathart.org/2022/bridges2022-309.html.

[5] Z. Bylinskii, *et al.* "Towards Better User Studies in Computer Graphics and Vision." *Foundations and Trends in Computer Graphics and Vision*, vol. 15, no. 3, 2023, pp. 201–252.

[6] S. De Toffoli and V. Giardino. "Forms and Roles of Diagrams in Knot Theory." *Erkenntnis*, vol. 79, no. 4, 2014, pp. 829–842.

[7] M. Dehn. "Über Kombinatorische Topologie." *Acta Mathematica*, Vol. 67, 1936, pp. 1213–1268.

[8] J. Hoste, M. Thistlethwaite and J. Weeks. "The First 1,701,936 Knots." *The Mathematical Intelligencer*, vol. 30, no. 4, 1998, pp. 33–48.

[9] L. H. Kauffman. "Gauss Codes, Quantum Groups and Ribbon Hopf Algebras." *Reviews in Mathematical Physics*, vol. 5, no. 4, 1993, pp. 735–773.

[10] L. H. Kauffman and S. Lambropoulou. "Hard Unknots and Collapsing Tangles." *Knot Theory and its Applications to Physics and Biology*, Trieste, Italy, May 11–29, 2009, pp. 1–62,

[11] C. Petronio and A. Zanellati. "Algorithmic Simplification of Knot Diagrams: New Moves and Experiments." *Journal of Knot Theory and Its Ramifications*, vol. 25, no. 10, 2016, p. 1650059.

[12] K. Reidemeister. *Knotentheorie.* Springer, Berlin, 1932.

[13] P. E. Santos, P. Cabalar and R. Casati. "The Knowledge of Knots: An Interdisciplinary Literature Review." *Spatial Cognition & Computation*, vol. 19, no. 4, 2019, pp. 334–358.

[14] R. G. Scharein. *Interactive Topological Drawing.* PhD Thesis, University of British Columbia, 1998.

[15] J. Sweller. "Cognitive Load Theory, Learning Difficulty, and Instructional Design." *Learning and Instruction*, vol. 4, no. 4, 1994, pp. 295–312.

[16] J. Sweller. "Element Interactivity and Intrinsic, Extraneous, and Germane Cognitive Load." *Educational Psychology Review*, vol. 22, 2010, pp. 123–138.

[17] R. E. Tuzun and A. S. Sikora. "Verification of the Jones Unknot Conjecture up to 22 Crossings." *Journal of Knot Theory and its Ramifications*, vol. 27, no. 3, 2018, pp. 1840009 (18 pp.).