Q-ART: A Scalable Base-4 Owen-Scrambling Algorithm

Abdalla G. M. Ahmed

Khartoum, Sudan

ARTICLE INFO

Keywords:
Owen scrambling
SZ sequences
quasi-Monte Carlo
low discrepancy

ABSTRACT

Owen scrambling is a standard randomization technique for common low-discrepancy sampling distributions used in quasi-Monte-Carlo applications such as rendering. ART Owen scrambling is an implementation of binary Owen scrambling using the Adaptive Regular Tiles (ART) hierarchical indexing model, while Q-ART is a more recently introduced extension to handle the base-4 case, covering the full 4! permutations while still maintaining the method's simple and elegant structure. We provide a more in-depth exposition of this Q-ART algorithm, discuss technical implementation details, and demonstrate its practical use in rendering through the Q-ART-based StART sampler.

1. Introduction

Low-discrepancy (LD) constructions, especially (t, m, s)-nets and (t, s)-sequences (Niederreiter, 1992), are increasingly becoming the primary source of sampling points in rendering applications (Pharr et al., 2023). A major challenge in rendering is that sample distributions must vary across the different dimensions of light transport paths, as well as across pixels in the image plane. Ideally, these samples should be independent or even negatively correlated for adjacent pixels. The complex structure of LD distributions makes classical randomization techniques such as toroidal shifts unsuitable. Instead, Owen (1995) scrambling has become the standard randomization technique for LD nets and sequences. It transforms a deterministic distribution into a randomized one while preserving the underlying net structure and its associated low-discrepancy properties.

For many decades, Sobol' (1967) sequences were essentially the only (t, s)-sequences used in graphics applications, and because these sequences are binary-base constructions, most research in graphics has focused exclusively on binary Owen scrambling. More recently, however, a few applications have emerged that require base-4 Owen scrambling. For example, the Z-indexed distribution of samples over pixels (Ahmed and Wonka, 2020) requires nested shuffling of the Z-order (Morton) indices (Morton, 1966) of pixels, which is essentially equivalent to base-4 Owen scrambling. In addition, the recently introduced SZ family of sequences (Ahmed et al., 2025) may include base-4 constituent dimensions, which can benefit from base-4 Owen scrambling as a richer alternative to the still-applicable binary variant. To address this growing demand for efficient high-quality base-4 Owen scrambling, Ahmed (2025) recently extended the binary ART-Owen scrambling algorithm (Ahmed et al., 2023) to the base-4 case. The resulting Q-ART algorithm has already been successfully integrated and tested within rendering stacks (Ahmed et al., 2025, 2026). The Q-ART concept has also recently been generalized to higher power-of-two bases (Ahmed and Wang, 2026). In this paper, we study Q-ART Owen scrambling in greater depth, covering implementation details and providing a more extensive evaluation.

2. Technical Background and Related Work

In this section we briefly review the essential technical background required for the discussion that follows, and discuss the most closely related work. We start with an intuitive definition of nets and (t, s)-sequences (Niederreiter, 1992) that encompasses the scope of applications considered in this paper. A (t, m, s)-net in base b is a set of b^m points in the s-dimensional unit hypercube such that, for every partition of the domain into b^{m-t} congruent hyperrectangular cells, each cell contains exactly b^t points. A (t, s)-sequence in base b is an infinite sequence of points in the s-dimensional unit hypercube such that, for every integer m, every consecutive block of b^m points form a (t, m, s)-net in base b.

2.1. Owen Scrambling

Owen (1995) showed that a valid (t, m, s)-net or (t, s)-sequence in base b can be obtained from an existing one by recursively scrambling the base-b digits of the sample coordinates along each axis. That is, viewing the unit interval as a base-b tree rooted at the fractional point '.', Owen scrambling randomly shuffles the branches below each node, and this scrambling, when applied to some or all axes, provably preserves the t parameter of the resulting (t, m, s)-net or (t, s)-sequence.

Implementing Owen scrambling, however, is far more challenging than explaining it, since it requires populating and querying a random-access tree of random permutations over the digits of the underlying base. The key requirement is that the permutations associated with ancestor nodes must be accessible to all descendant nodes, allowing them to be processed in arbitrary order.

abdalla_gafar@hotmail.com (A.G.M. Ahmed)
abdallagafar.com (A.G.M. Ahmed)
ORCID(s): 0000-0002-2348-6897 (A.G.M. Ahmed)

2.2. ART-Owen Scrambling

Ahmed et al. (2023) recently introduced ART-Owen scrambling, an elegant formulation of binary Owen scrambling that builds on the earlier grammar-based Adaptive Regular Tiles (ART) model (Ahmed et al., 2017) by augmenting the basic ART hierarchy with propagated scrambling data. The idea may be summarized as follows. A grammar, i.e., a set of production rules

$$(A, B, \ldots) \mapsto ((A, B), (E, C), \ldots)\,, \tag{1}$$

is defined over an alphabet

$$\Sigma = \{A, B, \ldots\} \tag{2}$$

of symbols of any prescribed size. The production rules are used to build a tree that aligns with the target scrambling tree. Each symbol, and thus each instance of it, is equipped with (i) a random scrambling bit and (ii) a vector of bits, one bit per level, used to xor-scramble the scrambling bits of descendant nodes. Even though all instances of each symbol generate identical subtrees thanks to the context-free grammar in Eq. (1), these trees still vary in their associated data due to inherited scrambling bits. Thus, the essence of ART-Owen scrambling is to impose some context awareness by passing information to descendants, thereby enriching the scrambling tree and further randomizing the resulting distribution.

The ART-Owen scrambling idea may be extended naively to the base-4 quadtree case by interpreting successive pairs of bits as base-4 digits. This, however, only covers four of the $4! = 24$ possible permutations of branches in a quadtree, leading to significantly poorer scrambling. This limitation is addressed by our proposed quaternary (base-4) ART (Q-ART) Owen scrambling, introduced in the following section.

3. Q-ART

The key observation behind Q-ART is that there are exactly six invertible 2×2 matrices over GF(2), the Galois field with two elements. Combined with the four possible translations, the set of 24 distinct permutations of base-4 digits can therefore be neatly factored into an affine transformation

$$X \mapsto AX+B, \quad X, B \in \left\{ \begin{smallmatrix}0\\0\end{smallmatrix}, \begin{smallmatrix}1\\0\end{smallmatrix}, \begin{smallmatrix}0\\1\end{smallmatrix}, \begin{smallmatrix}1\\1\end{smallmatrix} \right\}, \; A \in \left\{ \begin{smallmatrix}1&0\\0&1\end{smallmatrix}, \begin{smallmatrix}1&1\\0&1\end{smallmatrix}, \begin{smallmatrix}1&0\\1&1\end{smallmatrix}, \begin{smallmatrix}0&1\\1&0\end{smallmatrix}, \begin{smallmatrix}1&1\\1&0\end{smallmatrix}, \begin{smallmatrix}0&1\\1&1\end{smallmatrix} \right\}, \tag{3}$$

where gray and black dots represent zeros and ones, respectively. The essence of this decomposition is that, unlike the monolithic set of permutations, the affine representation admits efficient composition along the tree. Each symbol carries its own transformation matrix and translation vector, together with vectors of matrices and translations, one pair per level, that are propagated to descendants. The affine components at a node are obtained by concatenating ancestor contributions via efficient bitwise operations.

4. Implementation

In this section we discuss implementation details of the Q-ART algorithm. While the original paper introducing the algorithm (Ahmed, 2025) is already accompanied by implementation code, we have introduced several changes since then, which we highlight explicitly below.

4.1. Alphabet and Grammar

Among the various choices discussed in the original ART-Owen paper (Ahmed et al., 2023), we strongly advocate an alphabet and associated grammar derived from the Thue–Morse word T (Lothaire, 2002), thanks to their balanced symbol frequencies and repetition-avoidance properties. The construction has a single parameter, the *id length*: the alphabet is defined as the set of distinct subwords of T of that length. We omit the details of deriving the associated grammar here, and refer the interested reader to Ahmed et al. (2017, 2023) for the full derivation process.

We recommend an id length of 6, following the original implementation, which yields a power-of-two alphabet size of 16. Larger alphabets may also be considered, but this size performed well in our experiments. It is important to note that, depending on the application, the quadtree of base-4 Owen scrambling may be interpreted semantically either as quarters of a line segment, e.g., a single dimension of a $(0, 4)$ SZ sequence (Ahmed et al., 2025), or as quadrants of a 2D square, e.g., the image plane in Z-indexed sampling (Ahmed and Wonka, 2020). Furthermore, the repetition-avoidance mechanism of ART requires that 2D alphabets and grammars be constructed as Cartesian products of 1D alphabets. Consequently, the recommended alphabet size becomes 256 symbols for 2D scrambling.

4.2. Scrambling Vector

As mentioned earlier, the affine components of the permutation at a node are obtained by concatenating ancestor contributions. We observed little difference between concatenating or omitting the inherited matrix component. Therefore, we use a fixed matrix for each symbol across all instances and only scramble the translation bits over descendant nodes, thereby avoiding the computational cost of repeated matrix×matrix multiplications.

4.3. Encoding

The original implementation tabulates the matrices and xor vectors separately. A reasonable motivation for this is that the 16 matrices can be encoded compactly in a single 64-bit word. This approach, however, does not scale to the 2D case of 256 symbols.

We therefore adopt a more flexible encoding that combines the matrix and xor vector in a single 32-bit word, allowing the entire lookup table to fit neatly into a GPU-friendly 1KB table. In this setting there is no tangible advantage in the four-bit encoding of the matrices. Instead, we encode the actual permutation as pairs of bits that fit neatly in a single byte, replacing the matrix-vector multiplication with slightly less expensive shift-and-mask operations. With such direct encoding of permutations, instead of restricting ourselves to the six linear permutations generated by the matrices, we advocate selecting from the full set of 24 affine permutations. This performs equivalently while also

providing a table of random permutations that can be reused for other applications.

We place the permutation in the least significant byte and randomly populate the remaining bits with the xor vector. This encoding offers multiple advantages: (i) the semi-random permutation bits become reusable as translations, and (ii) the entire table can also serve as a random-number table. For example, Q-ART and ART-Owen scrambling may now share the same lookup table, which is especially helpful in GPU applications.

Aside from the permutation data, the remaining data required to implement the Q-ART algorithm are the production rules, which are invariant for a given alphabet size. Notably, we observed a significant GPU performance difference between storing these rules in lookup memory and inlining the packed table using the ternary selection operator.

4.4. Actual Code

Despite its apparent complexity, Q-ART reduces to only a few lines of actual implementation code. Specifically, the snippet

```
inline int getChildId1D(int id, int childNo) {
    #ifdef __CUDA_ARCH__
    return ((
        id < 8 ?
        (childNo ? 0xC1B97531 : 0x20A86420) :
        (childNo ? 0xBCF7FED9 : 0xA24646A8)
    ) >> ((id & 7) << 2)) & 15;
    #else
    return ((
        childNo ?
        0xBCF7FED9C1B97531llu :
        0xA24646A820A86420llu
    ) >> (id << 2)) & 15;
    #endif
}
```

(4)

shows our current implementation of the production rules. This retrieves the immediate child id in the binary ART production rule. Retrieving a child id in a 1D quadtree corresponds to retrieving the grandchild id by applying Eq. (4) twice, while retrieving a 2D child id is obtained as the Cartesian product of the corresponding 1D id components. Next, the excerpt

```
inline uint32_t qart(uint32_t xIn, int id) {
    uint32_t xorVector(0);
    uint32_t xOut(0);
    for (int bitNo = 0; bitNo < 16; bitNo++) {
        uint32_t scrmbl = qartTable[id];
        xorVector = (xorVector << 2) ^ scrmbl;
        int x = xIn >> 30;
        xIn <<= 2;
        x = (scrmbl >> (2 * x)) & 3;
        x ^= xorVector >> 30;
        xOut = (xOut << 2) | x;
        id = getChildId(id, x);
    }
    return xOut;
}
```

(5)

constitutes the complete implementation of the 1D case. Scrambling a 2D Z-index domain is similar, but splits each base-4 digit into a pair of quadrant coordinates.

4.5. Inversion

As in ART-Owen scrambling (Ahmed et al., 2023), an important advantage of Q-ART is that it is easily invertible, allowing the original point location to be recovered. This is useful in applications such as adaptive sampling. Indeed, unscrambling is almost identical to scrambling: it uses the same propagated translations, the same production rules, and inverse versions of the same digit permutations, with the elementary operations merely rearranged to undo the forward mapping. Notably, four of the six matrices are self-inverting, suggesting a simple way to handle them. For applications that require inversion, the original 4-bit matrix representation may become attractive, as the forward and inverse matrices can then be encoded compactly in a single byte.

5. Results

In this section, we evaluate the performance of Q-ART from different perspectives.

5.1. 1D Scrambling

In Figure 1, we compare variants of Q-ART scrambling to hash-based and full-tree implementations applied axis-wise to a base-4 Hammersley net. Visually, the resulting quality is indistinguishable from that of the full-tree implementation. Interestingly, the average discrepancy curves in Figure 2 show that both Q-ART and the hash-based implementation achieve slightly lower discrepancy than the full-tree reference, an unexpected observation that merits future investigation. Furthermore, although the multiplication of vectors of matrices propagated to descendants can be implemented efficiently, we observe little difference compared to fixing the matrices for each symbol, validating our earlier design choice.

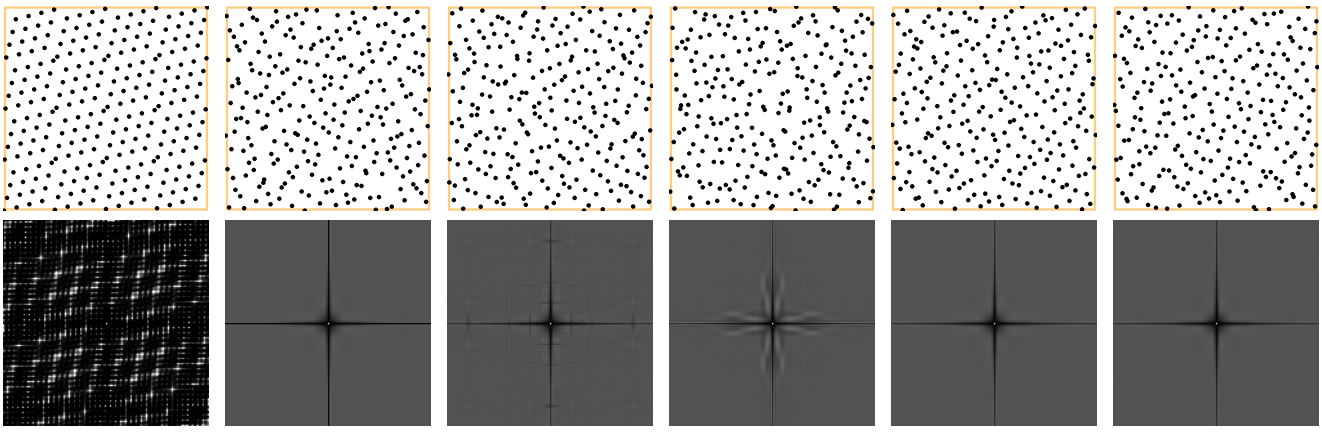
While it is difficult to notice a difference in the point layouts, the frequency power spectra reveal an evident distortion in the naive extension of ART-Owen scrambling to base 4 shown in Figure 1(d). A less severe but still noticeable distortion appears in the hash-based implementation of base-4 Owen scrambling used by Ahmed and Wonka (2020) for Z-Sampler, shown in Figure 1(c). Another subtle limitation of such hash-based scrambling is that the arithmetic hash function may become entangled with the algebraic constructions that generate the underlying point distributions, as we have actually observed in some of our experiments.

5.2. 2D Scrambling

Next, we assess the scrambling performance in 2D by applying Owen scrambling to a uniform 16×16 grid. In principle, base-4 Owen scrambling should transform such a grid into a perfect jittered grid, also known as a stratified point set. As demonstrated in Figure 3, the performance of Q-ART is close to this ideal target and clearly superior to hashing, whose frequency spectrum exhibits strong spikes that persist across randomized instances.

5.3. Rendering

Q-ART has successfully been integrated into actual rendering frameworks, playing a primary role in the StART sampler and an auxiliary role in several other samplers of the



(a) Input (b) Full Tree (c) Z-Hash (d) XOR Only (e) Fixed Matrix (f) Scrambled Matrix

Figure 1: Base-4 Owen scrambling of (a) a base-4 Hammersley net with 256 points, using: (b) a pair of full random scrambling trees for reference, (c) the Z-Hash algorithm of Ahmed and Wonka (2020), (d) base-2 ART-Owen scrambling (Ahmed et al., 2023), (e) our quaternary ART (Q-ART) Owen scrambling with fixed matrices, and (f) Q-ART Owen scrambling with scrambled matrices. The top row shows one point distribution, and the bottom row shows the power spectrum averaged over 1M instances.

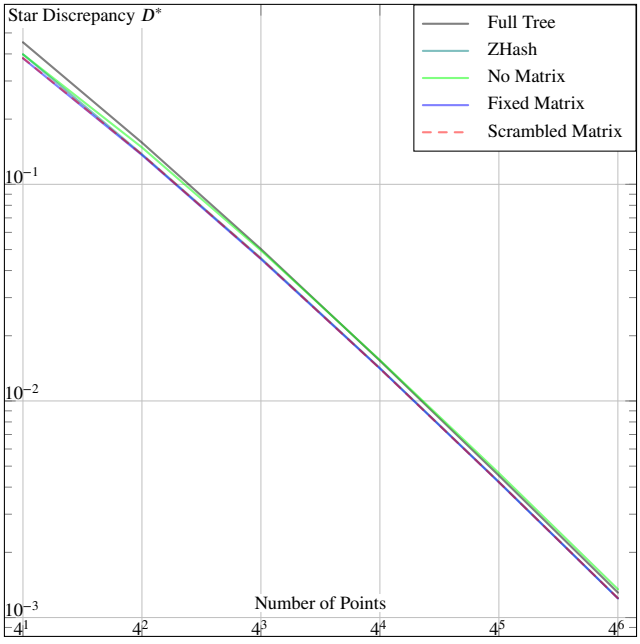
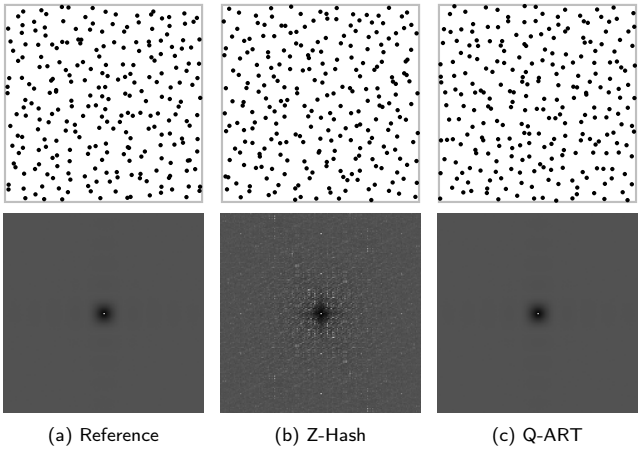


Figure 2: Discrepancy plots of Owen-scrambled base-4 Hammersley nets, averaged over 1000 realizations, benchmarking different Q-ART variants against full-tree and Z-Hash scrambling.

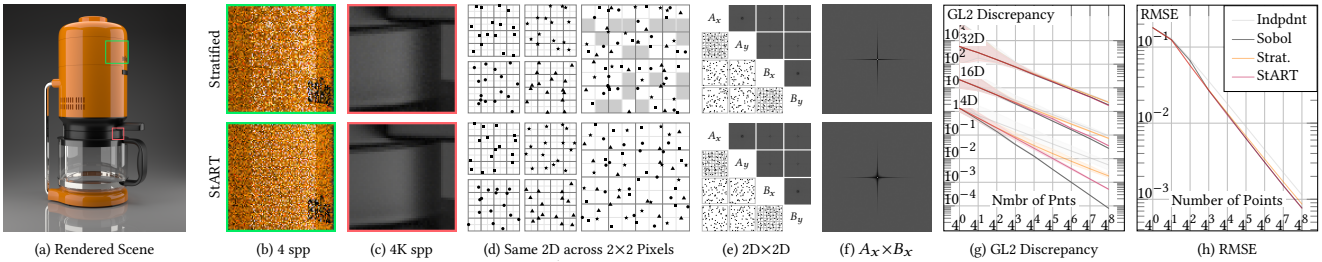


(a) Reference (b) Z-Hash (c) Q-ART

Figure 3: An instant 256-point distribution (top) and the frequency power spectrum averaged over 1M realizations (bottom), showing (a) a native jittered grid, obtained by placing each point randomly within its associated stratum, compared to Owen-scrambled regular grids using (b) the Z-Hash algorithm of Ahmed and Wonka (2020) and (c) Q-ART. Q-ART is visually indistinguishable from the native jittered grid, indicating excellent quasi-random scrambling quality. In contrast, hashing exhibits a noticeable spectral distortion.

recently introduced NILE meta-sampler framework (Ahmed et al., 2026). The core idea of NILE is to replace independently instantiated low-dimensional sampling distributions, such as stratified and blue-noise point sets, with carefully constructed sequences of mutually complementary instances of these distributions. These can then be evenly distributed over Z-indexed pixels, and systematically interleaved in higher-dimensional spaces using a binary high-dimensional low-discrepancy sequence such as Sobol or SZ for indexing.

This replacement amounts to a hierarchical indexing of binary intervals of the sampled domain, typically quadtrees for 2D domains, and occasionally involves base-4 Owen scrambling. For example, quaternary Owen scrambling of a sequence of Morton-indexed binary points in the unit square readily generates a nested sequence of stratified points, forming the core of the StART (Stratification with ART) sampler. A few other NILE-adapted distributions, such as blue noise and correlated multi-jitter, also employ base-4 Owen scrambling to shuffle and jitter the generated sequences. In all these cases, Q-ART provides an efficient GPU-friendly implementation. Indeed, the persistent frequency spikes exhibited by



(a) Rendered Scene (b) 4 spp (c) 4K spp (d) Same 2D across 2×2 Pixels (e) 2D×2D (f) $A_x \times B_x$ (g) GL2 Discrepancy (h) RMSE

Figure 4: Rendering and sample-space comparison of traditional stratified sampling and the NILE-orchestrated StART sampler, showing: (a) a reference rendering of the Coffee Maker scene, (b) close-up views at low sample rates, (c) close-up views at high sample rates, (d) distribution of domain samples across 2×2-pixel blocks, (e) spatial and spectral plots of two subspaces and their axis-wise pairings at 64 spp, (f) close-ups of spectral plots of axis-wise pairings at 4K spp, showing the effect of LD interleaving, (g) generalized L_2 discrepancy over different dimensions, and (h) RMSE convergence plots. Reproduced from the NILE paper (Ahmed et al., 2026, Fig. 6).

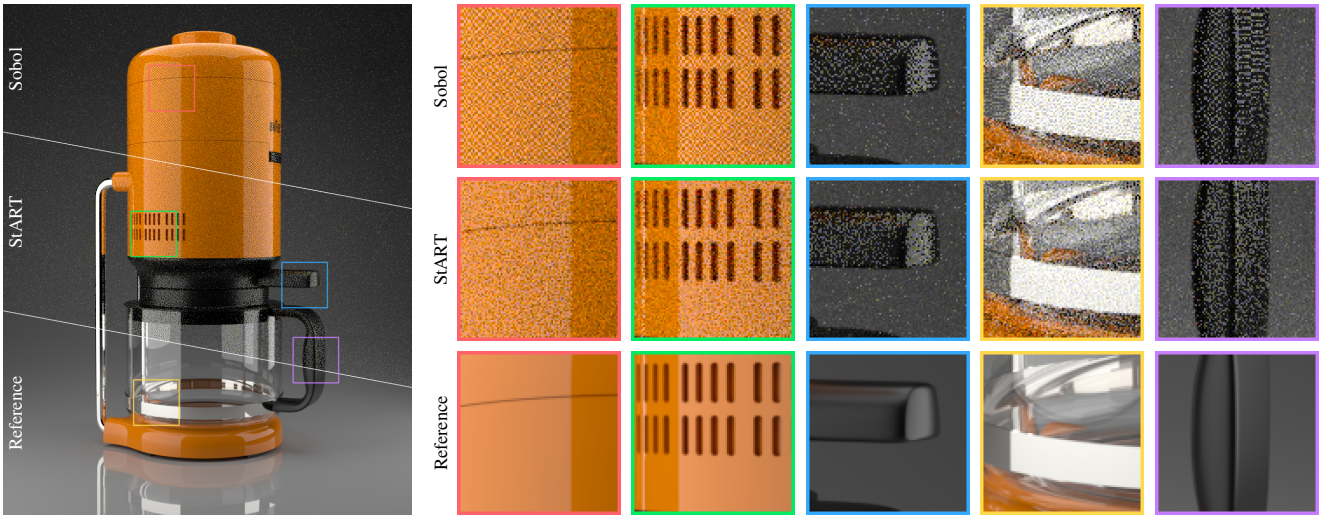


Figure 5: An example rendered scene comparing the visual quality of StART, a Q-ART-based sampler, to global Sobol, using 16 samples per pixel and a maximum path depth of 17. StART replaces the structured Monte Carlo noise artifacts visible in global Sobol sampling with more broadband noise, which is often visually less objectionable. Coffee Maker scene from Benedikt Bitterli's *Rendering resources* (Bitterli, 2016).

hash-based scrambling, as observed in Section 5.2, make it less suitable as a robust building block for rendering samplers. The full-table alternative, on the other hand, is also impractical for rendering applications, since scrambling N samples requires $\mathcal{O}(N)$ storage per dimension, whereas Q-ART replaces the full tree by a small GPU-friendly lookup table that is reusable across dimensions.

Figure 4, reproduced from the NILE paper (Ahmed et al., 2026), compares the StART sampler to its independently instantiated stratified counterpart, and benchmarks both against the well-established global Sobol sampler. In this and many other tested scenes, StART outperforms global Sobol in terms of convergence speed and visual quality, as demonstrated in Figure 4(h) and Figure 5, respectively, and consistently outperforms Halton in all tested scenes. These results provide solid evidence of the practical utility of StART and, in turn, of the underlying Q-ART implementation that enables its efficient base-4 Owen scrambling. Remarkably, this improvement in rendering quality is accompanied by multiple reductions in computational cost. For example, the measured overall GPU rendering speed of the StART sampler is within 5% of that of the Independent sampler, and is consistently faster than global Sobol. Moreover, using only a 1KB lookup table, the Q-ART-based StART sampler can cover 256 sample dimensions, whereas a comparable global Sobol table would require 256 matrices × 32 columns × 4 bytes, i.e., 32KB.

6. Conclusion

In this paper, we presented a detailed exposition of Q-ART, an efficient and scalable base-4 Owen scrambling algorithm, including an efficient implementation and experimentally validated parameter recommendations. A public implementation of ART-Owen, Q-ART, and related ART-Owen scrambling variants will be released in a dedicated Git repository. Compared to the two alternatives presented by Ahmed and Wonka (2020) for Z-indexed sampling, we showed that Q-ART improved substantially over hashing in

quality, while reducing the lookup-table size relative to their table-based solution from 64KB to only 1KB. We further demonstrated its practical relevance through its integration in the StART sampler of the NILE framework, where Q-ART serves as the efficient base-4 Owen scrambling mechanism in a rendering pipeline.

Declaration of Generative AI and AI-assisted Technologies in the Manuscript Preparation Process

During the preparation of this work, the author used ChatGPT by OpenAI for language editing, phrasing suggestions, LaTeX formatting assistance, and revision checking. The author reviewed and edited the output as needed and takes full responsibility for the content of the published article.

References

Ahmed, A.G.M., 2025. Q-ART Owen Scrambling, in: Computer Graphics and Visual Computing (CGVC), The Eurographics Association. doi:10.2312/cgvc.20251215.

Ahmed, A.G.M., Niese, T., Huang, H., Deussen, O., 2017. An Adaptive Point Sampler on a Regular Lattice. ACM Trans. Graph. 36. doi:10.1145/3072959.3073588.

Ahmed, A.G.M., Pharr, M., Ostromoukhov, V., Huang, H., 2025. SZ Sequences: Binary-Constructed $(0, 2^q)$-Sequences. ACM Trans. Graph. 44. doi:10.1145/3763272.

Ahmed, A.G.M., Pharr, M., Ostromoukhov, V., Huang, H., 2026. NILE: Nested Interleaving of Low-Dimensional Elements. ACM Trans. Graph. 45. doi:10.1145/3811283.

Ahmed, A.G.M., Pharr, M., Wonka, P., 2023. ART-Owen Scrambling. ACM Trans. Graph. 42. doi:10.1145/3618307.

Ahmed, A.G.M., Wang, C., 2026. Generalized ART-Owen Scrambling, in: Computer Graphics & Visual Computing (CGVC) 2026. doi:10.2312/cgvc.20261017.

Ahmed, A.G.M., Wonka, P., 2020. Screen-Space Blue-Noise Diffusion of Monte Carlo Sampling Error via Hierarchical Ordering of Pixels. ACM Trans. Graph. 39. doi:10.1145/3414685.3417881.

Bitterli, B., 2016. Rendering Resources. URL: https://benedikt-bitterli.me/resources/. accessed: 2025-11-08.

Lothaire, M., 2002. Algebraic Combinatorics on Words. Encyclopedia of Mathematics and its Applications, Cambridge University Press.

Morton, G.M., 1966. A Computer Oriented Geodetic Data Base; and a New Technique in File Sequencing .

Niederreiter, H., 1992. Random Number Generation and Quasi-Monte Carlo Methods. SIAM.

Owen, A.B., 1995. Randomly Permuted (t, m, s)-Nets and (t, s)-Sequences, in: Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, Springer New York. pp. 299–317.

Pharr, M., Jakob, W., Humphreys, G., 2023. Physically Based Rendering: From Theory to Implementation. 4th ed., MIT Press.

Sobol', I.M., 1967. On the Distribution of Points in a Cube and the Approximate Evaluation of Integrals. Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki 7, 784–802.