KANDLES Physical Block Demo — MVP Specification

Document Version: 0.1.0-draft
Last Updated: 2026-02-19
Status: Working Draft
Patent Reference: US 2024/0248922 A1, "System and Methods for Searching Text Utilizing Categorical Touch Inputs" (Merkur)

---

1. Executive Summary

A tangible demonstration of Mike Merkur's patented KANDLES system using physical colored blocks arranged in a 7×7 grid. Users encode text into colored blocks following the patent's phonetic-to-color mapping, then scan the arrangement to decode the original message.

The Goal: "Idiot proof" A→B demonstration

- **A** = Type text, build colored blocks
- **B** = Scan blocks, see text

No technical knowledge required. Physical blocks become the data medium — like a colorful, semantic QR code.

---

2. Physical Blocks as Scannable Data (The "Colored QR Code")

2.1 The Vision

Mike's core insight: **Physical toy blocks can encode digital data that anyone can scan and decode** — just like QR codes or Spotify codes, but:

- More visually interesting (colors vs black/white)
- Tactile and buildable (toy-like interaction)
- Semantically meaningful (colors = sounds = language)

2.2 How It Compares to QR Codes

Property	QR Code	KANDLES Blocks
Medium	Printed 2D pattern	Physical 3D blocks
Colors	Black & white only	7 semantic colors
Encoding	Binary data	Phonetic sounds
Interaction	Look at it	Build it, touch it, rearrange it
Error correction	Reed-Solomon built-in	Via CPSC (future)
Fun factor	Low (utilitarian)	High (toy-like)

Property	QR Code	KANDLES Blocks
Scannability	Any QR reader	Custom KANDLES app

2.3 Physical Stacking = Data Layering

Unlike flat QR codes, blocks can be **stacked vertically**:

```
        ┌─────┐
        │ L3  │   ← Layer 3 (Grid 3 data)
    ┌───┴─────┴───┐
    │     L2      │   ← Layer 2 (Grid 2 data)
┌───┴─────────────┴───┐
│         L1          │   ← Layer 1 (Grid 1 data / base)
└─────────────────────┘
```

Stacking Options:

1. **Flat Grid (MVP):** Single 7×7 layer, scan from above
   - Simplest to build and scan
   - ~17 characters capacity
2. **Stacked Grids:** Multiple 7×7 layers stacked vertically
   - Scan each layer separately (peel apart)
   - Or scan exposed "face" of the stack (front/side view)
   - Extended message capacity
3. **3D Sculpture:** Freeform block arrangement
   - Advanced: encode data in the 3D structure itself
   - Multiple scannable faces
   - Future capability (not MVP)

2.4 Demo Positioning

Elevator Pitch:

> "You know QR codes? Imagine if instead of printing a pattern, you *built* it with colored blocks. Then anyone can scan it and read your message. That's what this does — but it's fun, it's physical, and the colors actually mean something."

Key Differentiators from QR:

- **Buildable:** User physically constructs the code
- **Colorful:** 7 colors vs binary black/white
- **Semantic:** Colors map to language sounds (not arbitrary)
- **Tactile:** Hands-on toy experience
- **Stackable:** 3D data encoding potential

3. The KANDLES System (Patent Foundation)

2.1 Color-Number-Sound Mapping

Per US 2024/0248922 A1 (FIG. 20), the patent defines a **fixed color sound sequencing** algorithm called KANDLES:

Number	First Consonant Sounds	Color	Nature Symbol	Mnemonic
1	K, G, J, Ch	Yellow	Sun	**K**andles
2	M, N	Gray	Moon	**M**oon
3	T, D, Th	Red	Fire	**T**orch
4	R, L	Blue	Water	**R**iver
5	Y, W, H, Kh	Green	Tree	**W**ood
6	P, B, F, V	Purple	Flower	**P**etal
7	S, Z, Sh	Brown	Soil	**S**oil

Key Rule: Map the **first consonant sound** of each word to its corresponding color. Vowels at word beginnings are typically skipped or handled specially.

2.2 Grid Structure

Per the patent drawings (FIG. 3-6):

- **7 columns × 7 rows** = 49 cells
- Each cell contains one color (or black for unused/padding)
- Grid has equal number of columns and rows
- Colors correspond to encoded text following KANDLES mapping

2.3 Encoding Capacity

- 7 colors = $\log_2(7) \approx 2.8$ bits per cell
- 49 cells × 2.8 bits ≈ **137 bits per grid**
- Practical capacity: ~17 characters or short phrase per grid
- Longer messages: stack/sequence multiple grids

4. Demo Kit — Bill of Materials

3.1 Physical Components

Item	Quantity	Notes
Yellow blocks	10	KANDLES #1 (K, G, J, Ch sounds)
Gray blocks	10	KANDLES #2 (M, N sounds)

Item	Quantity	Notes
Red blocks	10	KANDLES #3 (T, D, Th sounds)
Blue blocks	10	KANDLES #4 (R, L sounds)
Green blocks	10	KANDLES #5 (Y, W, H sounds)
Purple blocks	10	KANDLES #6 (P, B, F, V sounds)
Brown blocks	10	KANDLES #7 (S, Z, Sh sounds)
Black blocks	20	Padding/unused cells
Baseplate	1	7×7 grid with cell markings

Recommended Block Type: Mega Bloks or similar 1"+ blocks with flat, solid colors. Avoid patterns, gradients, or reflective surfaces.

3.2 Software Components (MVP)

Component	Platform	Purpose
Encoder App	Mobile (iOS/Android) or Web	Text → KANDLES grid instructions
Decoder App	Mobile with camera	Camera → color grid → text

Can be combined into single app with two modes.

---

5. Encoding Process

4.1 Algorithm: Text to Color Grid

```
INPUT: Text string (e.g., "HELLO WORLD")
OUTPUT: 7×7 color grid

1. Tokenize input into words
2. For each word:
   a. Extract first consonant sound
   b. Map to KANDLES number (1-7)
   c. Map number to color
3. Fill grid left-to-right, top-to-bottom
4. Pad remaining cells with black
```

4.2 Worked Example: "THE CAT"

Word	First Sound	KANDLES #	Color
THE	Th	3	Red
CAT	C (hard K)	1	Yellow

Grid Output (simplified):

```
[Red] [Yellow] [Black] [Black] [Black] [Black] [Black]
[Black] [Black] [Black] [Black] [Black] [Black] [Black]
... (remaining rows black)
```

4.3 Handling Edge Cases

Case	Handling
Word starts with vowel	Use first consonant after vowel, or encode as special marker
Numbers in text	Spell out ("3" → "THREE" → Red)
Punctuation	Ignore
Unknown characters	Skip or error

---

6. Decoding Process

5.1 Algorithm: Color Grid to Text

```
INPUT: Camera image of 7×7 block grid
OUTPUT: Decoded text

1. Detect grid boundaries in image
2. Segment into 7×7 cells
3. For each cell:
   a. Classify color (Yellow/Gray/Red/Blue/Green/Purple/Brown/Black)
   b. Map to KANDLES number
   c. Map to sound group
4. Reconstruct words from sound sequence
5. Display decoded text
```

5.2 Color Detection Requirements

For reliable detection, ensure:

- **Consistent lighting** (avoid harsh shadows)
- **Distinct colors** (the 7 KANDLES colors have good separation)
- **Clean blocks** (no dirt, stickers, or mixed colors)
- **Flat capture angle** (camera perpendicular to grid)

5.3 Detection Confidence

MVP approach:

- Require >90% confidence per cell

- Flag low-confidence cells for user review
- Allow manual correction before final decode

---

7. Mobile App Specification (MVP)

6.1 Screens

Screen 1: Home

- Two buttons: "Encode" and "Decode"
- Brief instructions

Screen 2: Encode

- Text input field
- "Generate Grid" button
- Output: 7×7 visual grid showing which color block goes where
- Cell labels: "Row 1, Col 3: Place YELLOW block"
- Optional: Step-by-step guided mode

Screen 3: Decode

- Camera viewfinder with 7×7 overlay grid
- "Capture" button
- Processing indicator
- Output: Decoded text display
- Confidence indicator per cell

6.2 Technical Stack (Suggested)

Layer	Technology	Notes
Mobile Framework	React Native or Flutter	Cross-platform
Camera/Vision	OpenCV or ML Kit	Color classification
KANDLES Logic	TypeScript/Dart	Encode/decode algorithms
UI Components	Standard mobile UI	Grid display, camera

6.3 MVP Scope Boundaries

In Scope:

- Single 7×7 grid encode/decode
- English text input
- 7-color KANDLES mapping
- Basic camera capture and color detection

Out of Scope (v1):

- Multi-grid sequences

- Non-English languages
- Real-time video decoding
- Cloud storage/sharing
- Error correction (see §10 CPSC section)

---

8. Demo Scenarios

7.1 Investor Pitch Demo

Setup: Blocks pre-sorted by color, baseplate ready, app installed on two phones

Script:

1. "I'm going to send you a secret message using only colored blocks."
2. [Type message into Phone A encoder]
3. "The app tells me which colors to place where." [Build grid]
4. "Now you scan it with your phone." [Hand Phone B to investor]
5. [Investor scans] → Message appears
6. "Your colored blocks just transmitted data. This is protected by US Patent 2024/0248922."

Duration: 2-3 minutes

7.2 Educational/Kid Demo

Setup: Pre-made instruction card with child's name encoded

Script:

1. "Can you build this pattern?" [Show color grid card]
2. [Child places blocks]
3. "Now let's see what it says!" [Scan]
4. Child's name appears
5. "You just sent a secret message with toys!"

7.3 Trade Show / Booth Demo

Setup: Large display showing encode screen, physical blocks on table

Flow:

1. Visitor types their name
2. Display shows grid pattern
3. Visitor builds it (or staff assists)
4. Visitor scans with their own phone (app or web link)
5. Name appears — visitor keeps the photo as souvenir

---

9. Implementation Roadmap

Phase 1: Proof of Concept (1-2 weeks)

- [ ] KANDLES encode/decode logic in Python or JavaScript
- [ ] Command-line tool: text → grid image
- [ ] Procure physical blocks (7 colors + black)
- [ ] Manual test: encode, build, photograph, manually verify

Phase 2: Basic App (2-3 weeks)

- [ ] Mobile app shell with Encode/Decode screens
- [ ] Grid visualization for encoder output
- [ ] Static image color detection (not real-time)
- [ ] End-to-end test with physical blocks

Phase 3: Demo Ready (1-2 weeks)

- [ ] UI polish and guided instructions
- [ ] Improve color detection accuracy
- [ ] Create demo script and materials
- [ ] Record demo video (60-90 seconds)

Phase 4: Refinement (Ongoing)

- [ ] Real-time camera preview with grid overlay
- [ ] Multi-grid support for longer messages
- [ ] Error detection/correction (see §10 CPSC Integration)
- [ ] Sharing/export features

---

10. CPSC Integration (Future Enhancement)

This section describes how **Constraint-Projected State Computing (CPSC)** can enhance the KANDLES block system beyond the base patent capabilities.

9.1 What Mike's Patent Covers (Complete Without CPSC)

Feature	Status
Color-to-phonetic mapping	☑ Fully specified
7×7 grid generation	☑ Fully specified
Text encoding/decoding	☑ Fully specified
Touch-based categorical search	☑ Fully specified

For the MVP demo, Mike's patent is self-sufficient.

9.2 What CPSC Adds

9.2.1 Error Recovery via Constraint Projection

Mike's patent assumes **perfect input** — the camera sees exactly what was placed. Real-world problems:

- Lighting changes color perception
- Block is slightly rotated/misaligned
- One block is missing or wrong color
- Camera blur or focus issues

CPSC Solution: Project valid states from degraded input using constraint satisfaction.

```
Example:
Scanned grid has ambiguous cell at (3,4) — could be Blue or Purple
CPSC constraint: "HELLO" phonetic pattern requires Blue (L-sound) at this position
→ Resolves ambiguity via constraint projection
→ Returns correct decode despite imperfect scan
```

9.2.2 Increased Data Density

System	Bits per Cell	7×7 Grid Capacity
KANDLES (base patent)	~2.8 bits	~137 bits (~17 chars)
KANDLES + CPSC	~4-5 bits	~200-245 bits (~25-30 chars)

How: CPSC encodes state relationships between adjacent cells as constraints, enabling:

- Redundancy for error correction *without* reducing payload
- Inter-cell dependencies that increase effective information density

9.2.3 Physical Degradation Tolerance

Real blocks get scratched, faded, dirty over time. CPSC handles:

- Partial color information ("80% likely Green")
- Probabilistic state recovery
- Graceful degradation instead of decode failure

9.2.4 Multi-Grid Continuity

When messages span multiple 7×7 grids, CPSC maintains **cross-grid constraints:**

- Grid 2 must be consistent with Grid 1's ending state
- Detects if grids are assembled out of order
- Can recover from a missing middle grid in a sequence

9.3 CPSC Integration Architecture

```
┌─────────────────────────────────────────────────────────┐
│                    Application Layer                    │
│                 (Mike's KANDLES Patent)                 │
│    Text → Color Mapping → Grid Display → Touch Search   │
├─────────────────────────────────────────────────────────┤
```

```
│                    Computation Layer                     │
│                         (CPSC)                           │
│   Constraint Projection → Error Recovery → Validation    │
└──────────────────────────────────────────────────────────┘
```

Pitch framing:

> "KANDLES defines *what* to encode. CPSC defines *how* to robustly compute and recover it."

9.4 CPSC Feature Comparison

Feature	Mike's Patent Alone	With CPSC
Basic encode/decode	✅ Complete	✅ Same
Touch-based search	✅ Complete	✅ Same
Error correction	❌ Not addressed	✅ **Added**
Noisy/degraded input	❌ Fails	✅ **Recovers**
Higher data density	❌ Fixed ~2.8 bits	✅ **~4-5 bits**
Multi-block sequences	⚠️ Basic	✅ **Validated**
IP defensibility	✅ Patent protected	✅ **Stacked protection**

9.5 Recommendation

For MVP Demo: Use Mike's patent only. Keep it simple. The "idiot proof A→B" story is cleaner.

For Production Product: Integrate CPSC for:

- Robustness in real-world scanning conditions
- Technical moat that competitors can't easily replicate
- Patent stacking: application layer (Merkur) + computation layer (CPSC)

---

11. Open Questions

- [ ] Exact block dimensions and brand to standardize on?
- [ ] Web app vs native mobile app for MVP?
- [ ] Should grid cells be numbered/labeled on baseplate?
- [ ] How to handle words that start with vowels consistently?
- [ ] Demo video: professional production or phone-recorded?
- [ ] Target date for first working demo?

---

12. References

1. **US 2024/0248922 A1** — "System and Methods for Searching Text Utilizing Categorical Touch Inputs," Merkur, Filed Jan. 19, 2024

2. **1394.003US Formal Drawings** — Patent figures showing grid layouts
3. **CPSC Specification** — See `/docs/specification/CPSC-Specification.md`

Document History

Version	Date	Author	Changes
0.1.0	2026-02-19	Draft	Initial MVP specification
