Mini Vinci Formal Specification

This document formally specifies Mini Vinci and will be used for implementation. Document will talk about;

- Task
- Canvas
- Moves
- Instruction Set Language
- Scoring

Task

The task of the participant is to take a `canvas`, apply various `moves` over the canvas using the `Instruction Set Language`, and receive as high `Score` as possible.

Canvas

Canvas is a 2-dimensional space, it's an abstract representation of a painting. Although there are many possible interpretations of a canvas, we will be using a set of hierarchical blocks in tree structure. A canvas will have a base block, which will have many child blocks generated by moves.

Moves

Below, we first provide the move taxonomy. Descriptions of each move is given below.

- Moves
    - Block Manipulator Moves
        - Coloring Moves
            - Absolute Coloring Move
        - Swap Move
    - Block Generator Moves
        - Cut Moves
            - 4-Way/Point Cut Move
            - 2-Way/Line Cut Moves
                - Vertical Line Cut Move
                - Horizontal Line Cut Move
        - Merge Move

Block Manipulator Moves

These moves manipulate the contents of existing blocks.

Coloring Moves

These moves change the color of a given block.

Absolute Coloring Moves

This coloring move simply colors a given block with the given RGB values.

Swap Move

This move changes the contents of two blocks.

Block Generator Moves

These moves create new blocks with new block ids.

Cut Moves

These moves create new sub-blocks from existing blocks. They extent the block-id of their parent block.

When a block with block-id `n` is cut, it generates sub-blocks such as `n.0, n.1, n.2`.

4-Way/Point Cut Move

For this move, the user picks a block, and a point inside. That block is then cut into 4 sub-blocks. These sub-blocks are numbered as `parent-block-id.0, parent-block-id.1, parent-block-id.2, parent-block-id.3` from bottom-left sub-block by counting counter-clockwise.

2-Way/Line Cut Moves

For these moves, the user picks a block, an orientation and a line coordinate inside.

Vertical Line Cut Move

For this move, the user picks a block, and a coordinate on the `x-axis`. The block is cut into two sub-blocks, numbered `parent-block-id.0, parent-block-id.1` from left to right.

Horizontal Line Cut Move

For this move, the user picks a block, and a coordinate on the `y-axis`. The block is cut into two sub-blocks, numbered `parent-block-id.0, parent-block-id.1` from bottom to top.

Merge Move

For this move, the user picks two adjacent compatible blocks. Two blocks are compatible if their merge yields a rectangle. The newly created block is numbered by incrementing a global block counter and naming the new block with that number. As an example, if we start from block 0, cut it into two pieces `0.1, 0.2`; merge those parts, we will end up with block-id `1`.

Instruction Set Language

ISL is a language for expressing moves over a canvas. ISL code is compiled into moves, which are then interpreted for processing the canvas.

A separate specification of ISL is given in ISLSpec.md file.

Scoring

The score is calculated by combining a similarity function and a cost function.

The similarity function is presented by the problem. Each problem comes with a target painting. Processed canvas is rendered for pixel similarity. Similarity is computed via **[Provided Color Difference Function]**

The cost function is presented by the moves. Each move has an associated cost. Move costs are decided upon the given tenets.

- The cost of moves are inverse log proportional to the size of the blocks. Smaller blocks yield higher costs.
- Cost(Merge) < Cost(Swap) < Cost(Cut) < Cost(Color) for the same sized blocks.
- No strategy using too small or too large blocks should be able to have high scores. We aim for medium sized blocks and a high number of merge+swap operations.

The actual functions will need to be decided after a series of experiments.