Pavemat

v0.1.0 2024-7-27 MIT

QuadnucYard

https://github.com/QuadnucYard/pavemat

The pavemat is a versatile tool for creating styled matrices with custom paths, strokes, and fills. It allows users to define how paths should be drawn through the matrix, apply different strokes to these paths, and fill specific cells with various colors. This function is particularly useful for visualizing complex data structures, mathematical matrices, and creating custom grid layouts.

Part I.
Basics

I.1. Pave string

A pave string defines paths through the matrix. This string consists of directional characters that guide the path from one cell to another. It can also include styled segments and custom control characters for more advanced configurations.

I.1.1. Basic Directional Characters

A pave string primarily uses the following basic directional characters to move through the matrix:

- `W`, `w`: Move up
- `S`, `s`: Move down
- `A`, `a`: Move left
- `D`, `d`: Move right

If a lowercase letter is used, this segment will not be displayed.

I.1.2. Styled Segments

To apply styles to specific segments of the path, you can enclose the style specifications in parentheses. These styles, **which affect only the subsequent segment until the closing parenthesis or the end of the segment**, can include changes to stroke properties such as dash pattern, color, and thickness. The style is evaluated as dictionary and then passed to `grid.hline.stroke` and `grid.vline.stroke`. **Styles can be overlaid.**

Examples:

- Solid Stroke: `"SS(dash: 'solid')DDD"`
- Colored Stroke: `"SS(paint: red)DDD"`
- Thickness and Dash: `` `SS(thickness: 2pt, dash: "dotted")DDD`.text ``

Single quotes (`'`) in the string will be simply replaced by double quotes (`"`), so there is no need to escape them.

In the example `"SS(dash: 'solid')DDD"`, the path first moves down twice and then uses a solid stroke for the next segment that moves right three times.

The style has its scope, ended by a right bracket `]`. You can optionally add one `[` after style parenthesis `)`, which is just ignored.

Examples:

```
"AA(paint: red, thickness: 2pt)WdD(paint: blue)Ww(thickness: 1pt, dash:
'dotted')AaS]Aw]W]D"
```

Explanations:

- `AA`:
  - Moves left twice (`AA`).
- `(paint: red, thickness: 2pt)WdD`
  - The stroke is red and 2pt thick.
  - Moves up (`W`), then down (`d`, hidden), and right (`D`).
- `(paint: blue)Ww`:
  - The stroke changes to blue, while thickness is still 2pt.
  - Moves up (`W`) and up again(`w`, hidden).
- `(thickness: 1pt, dash: 'dotted')AaS]`:
  - The stroke changes to 1pt thick with a dotted pattern.
  - Moves left (`A`), left (`a`), and down (`S`).
  - When encountering `]`, the style resets to the previous state (in this case, blue stroke).

I.1.3. Custom Direction Characters

If you prefer `UDLR` to `WASD`, you can customize control characters in through arguments. See usage.

I.2. Position

A position string is formatted as `"{x}-{y}"` or `"[{x}-{y}]"`. Examples include `"0-1"`, `"top-left"`, `"bottom-2"`, and `"[2-right]"`.

- `{x}`: Can be an integer or the literal `top` or `bottom`.
- `{y}`: Can be an integer or the literal `left` or `right`.

In the `fills` parameter, position strings determine which cells to fill with the specified color. By default, a position string without brackets (`"{x}-{y}"`) fills the connected block of cells containing `(x, y)`. When enclosed in brackets (`"[{x}-{y}]"`), it only fills the specified single cell.

Part II.
Usage

```
#pavemat(
  ⟨eq⟩,
  ⟨pave⟩: (),
  ⟨stroke⟩: (),
  ⟨fills⟩: (:),
  ⟨dir-chars⟩: (:),
  ⟨display-style⟩: true,
  ⟨block⟩: auto,
  ⟨debug⟩: false
)
```

The pavemat function in Typst allows for the creation of styled and filled matrices with custom paths, strokes, and fills.

Argument

⟨eq⟩ `math.equation` | `math.mat` | `array`

The input matrix expression to be styled. It can be a mathematical equation or a matrix. Specifically,

- A `math.equation`. It should contains only a `math.mat` as its body. Example: `$mat(1, 2; 3, 4)$`. `math.display` in the equation is not supported yet.
- A `math.mat`. Example: `math.mat((1, 2), (3, 4))`.
- A nested array. Example: `((1, 2), (3, 4))`.

If a matrix type is given, pavemat will use its style, including `row-gap`, `column-gap` and `delim`.

Argument

⟨pave⟩: () `str` | `dictionary` | `array`

Describes the pavement lines. It accepts the following formats:

- A path string like `"WASD"`.
- A dictionary with fields: `path`, `from` (optional, default: `"top-left"`), `stroke` (optional, default: empty). The path is a pave string.
- An array whose item type is either string or dictionary described above.

Argument

⟨stroke⟩ `length` | `color` | `gradient` | `pattern` | `dictionary`

The global stroke style applying to all segments. This argument will be passed to `cell.stroke`.

Accepts anything can be use as stroke. Examples: `blue + 1pt`, `(dash: "dashed", thickness: 0.5pt)`.

Argument

⟨fills⟩ color | gradient | pattern

Specifies the fill colors for specific cells. The key represents a position, and the value is the color passed to `cell.fill`. An empty key `""` is used for global fill.

Argument

⟨dir-chars⟩: (:) dictionary

Controls whether the output is in display style. Its fields will override the default `(up: "W", down: "S", left: "A", right: "D")`.

Example: `(up: "U", down: "D", left: "L", right: "R")`

Argument

⟨block⟩: auto auto | bool

Controls whether the output is block-style. If set to auto, it uses the block setting of the input equation.

Argument

⟨delim⟩: auto auto | str

The delimiter of the matrix. If set to auto, it uses the delimiter of the input matrix.

Argument

⟨display-style⟩: true bool

Controls whether the output is in display style. If set to `true`, the result will be granted a `math.display(...)`.

Argument

⟨debug⟩: false bool

Enables debug mode to show hidden lines. If set to a non-boolean value, it defines the debug stroke style.

Example:

```
#pavemat(
  $ mat(1, 2, 3; 4, 5, 6; 7, 8, 9; 10, 11, 12) $,
  pave: "dSDSDSLLAAWASSDD",
  fills: (
    "1-1": red.transparentize(80%),
    "1-2": blue.transparentize(80%),
    "3-0": green.transparentize(80%),
  ),
)
```