Draft for Information Only

Content

Manim VMobject

`VMobject` is a Vectorized Mobject. Vector graphics techniques are used to generate 2D graphics and 3D rendering in addition to a ordinary Mobject. A `VMobject` object is the key element used in Manim as a dummy vmobject container with base vmobject manipulating functions. A `VMobject` object focuses only on the internal structural design of a `VMobject` object.

Codes in Mobject.types.vectorized_mobject.py

Available codes defined in manimlib.mobject.types.vectorized_mobject.py

```
import itertools as it
import sys

from colour import Color

from manimlib.constants import *
from manimlib.mobject.mobject import Mobject
from manimlib.mobject.three_d_utils import get_3d_vmob_gradient
from manimlib.utils.bezier import bezier
from manimlib.utils.bezier import get_smooth_handle_points
from manimlib.utils.bezier import interpolate
from manimlib.utils.bezier import integer_interpolate
```

Five classes, `VMobject`, `VGroup`, `VectorizedPoint`, `CurvesAsSubmobjects`, and `DashedVMobject` are defined.

Import

The import defined in manimlib.mobject.types.vectorized_mobject.py:

```
import itertools as it
import sys

from colour import Color

from manimlib.constants import *
from manimlib.mobject.mobject import Mobject
from manimlib.mobject.three_d_utils import get_3d_vmob_gradient_start_and_end_points
from manimlib.utils.bezier import bezier
from manimlib.utils.bezier import get_smooth_handle_points
from manimlib.utils.bezier import interpolate
from manimlib.utils.bezier import integer_interpolate
from manimlib.utils.bezier import partial_bezier_points
from manimlib.utils.color import color_to_rgba
from manimlib.utils.iterables import make_even
from manimlib.utils.iterables import stretch_array_to_length
from manimlib.utils.iterables import tuplify
from manimlib.utils.simple_functions import clip_in_place
```

```
from manimlib.utils.space_ops import rotate_vector
from manimlib.utils.space_ops import get_norm
```

Class VMobject(Mobject)

class manimlib.mobject.types.vectorized_mobject.VMobject(Mobject)

version
19Dec2019

Configuration of VMobject

The configuration of a `VMobject` is defined in manimlib.mobject.types.vectorized_mobject.py

```
CONFIG = {
    "fill_color": None,
    "fill_opacity": 0.0,
    "stroke_color": None,
    "stroke_opacity": 1.0,
    "stroke_width": DEFAULT_STROKE_WIDTH,
    # The purpose of background stroke is to have
    # something that won't overlap the fill, e.g.
    # For text against some textured background
    "background_stroke_color": BLACK,
    "background_stroke_opacity": 1.0,
    "background_stroke_width": 0,
    # When a color c is set, there will be a second color
    # computed based on interpolating c to WHITE by with
    # sheen_factor, and the display will gradient to this
    # secondary color in the direction of sheen_direction.
    "sheen_factor": 0.0,
    "sheen_direction": UL,
    # Indicates that it will not be displayed, but
    # that it should count in parent mobject's path
    "close_new_points": False,
    "pre_function_handle_to_anchor_scale_factor": 0.01,
    "make_smooth_after_applying_functions": False,
    "background_image_file": None,
```

```
            "shade_in_3d": False,
            # This is within a pixel
            # TODO, do we care about accounting for
            # varying zoom levels?
            "tolerance_for_point_equality": 1e-6,
            "n_points_per_cubic_curve": 4,
        }
```

Functions

Functions defined in class `Mobject` are

- def get_group_class(self)
- # Colors
- def init_colors(self)
- def generate_rgbas_array(self, color, opacity)
- def update_rgbas_array(self, array_name, color=None, opacity=None)
- def set_fill(self, color=None, opacity=None, family=True)
- def set_stroke(self, color=None, width=None, opacity=None, background=False, family=True)
- def set_background_stroke(self, **kwargs)
- def set_style(self, fill_color=None, fill_opacity=None, stroke_color=None, stroke_width=None, stroke_opacity=None, background_stroke_color=None, background_stroke_width=None, background_stroke_opacity=None, sheen_factor=None, sheen_direction=None, background_image_file=None, family=True)
- def get_style(self)
- def match_style(self, vmobject, family=True)
- def set_color(self, color, family=True)
- def set_opacity(self, opacity, family=True)
- def fade(self, darkness=0.5, family=True)

- def get_fill_rgbas(self)
- def get_fill_color(self)
- def get_fill_opacity(self)
- def get_fill_colors(self)
- def get_fill_opacities(self)
- def get_stroke_rgbas(self, background=False)
- def get_stroke_color(self, background=False)
- def get_stroke_width(self, background=False)
- def get_stroke_opacity(self, background=False)
- def get_stroke_colors(self, background=False)
- def get_stroke_opacities(self, background=False)
- def get_color(self)
- def set_sheen_direction(self, direction, family=True)
- def set_sheen(self, factor, direction=None, family=True)
- def get_sheen_direction(self)
- def get_sheen_factor(self)
- def get_gradient_start_and_end_points(self)
- def color_using_background_image(self, background_image_file)
- def get_background_image_file(self)
- def match_background_image_file(self, vmobject)
- def set_shade_in_3d(self, value=True, z_index_as_group=False)
- # Points
- def set_points(self, points)
- def get_points(self)
- def set_anchors_and_handles(self, anchors1, handles1, handles2, anchors2)
- def clear_points(self)
- def append_points(self, new_points)
- def start_new_path(self, point)

- def add_cubic_bezier_curve(self, anchor1, handle1, handle2, anchor2)
- def add_cubic_bezier_curve_to(self, handle1, handle2, anchor)
- def add_line_to(self, point)
- def add_smooth_curve_to(self, *points)
- def has_new_path_started(self)
- def get_last_point(self)
- def is_closed(self)
- def add_points_as_corners(self, points)
- def set_points_as_corners(self, points)
- def set_points_smoothly(self, points)
- def change_anchor_mode(self, mode)
- def make_smooth(self)
- def make_jagged(self)
- def add_subpath(self, points)
- def append_vectorized_mobject(self, vectorized_mobject)
- def apply_function(self, function)
- def scale_handle_to_anchor_distances(self, factor)
- #
- def consider_points_equals(self, p0, p1)
- # Information about line
- def get_cubic_bezier_tuples_from_points(self, points)
- def get_cubic_bezier_tuples(self)
- def get_subpaths_from_points(self, points)
- def get_subpaths(self)
- def get_nth_curve_points(self, n)
- def get_nth_curve_function(self, n)
- def get_num_curves(self)
- def point_from_proportion(self, alpha)

- def get_anchors_and_handles(self)
- def get_start_anchors(self)
- def get_end_anchors(self)
- def get_anchors(self)
- def get_points_defining_boundary(self)
- def get_arc_length(self, n_sample_points=None)
- # Alignment
- def align_points(self, vmobject)
  - def get_nth_subpath(path_list, n)
- def insert_n_curves(self, n)
- def insert_n_curves_to_point_list(self, n, points)
- def align_rgbas(self, vmobject)
- def get_point_mobject(self, center=None)
- def interpolate_color(self, mobject1, mobject2, alpha)
- def pointwise_become_partial(self, vmobject, a, b)
- def get_subcurve(self, a, b)

Class VGroup(VMobject)

class manimlib.mobject.types.vectorized_mobject.VGroup(VMobject)

version
19Dec2019

Functions

Functions defined in class `VGroup` are

- def __init__(self, *vmobjects, **kwargs)

Class VectorizedPoint(VMobject)

class manimlib.mobject.types.vectorized_mobject.VectorizedPoint(VMobject) version 19Dec2019

Configuration

The configuration is defined in manimlib.mobject.types.vectorized_mobject.py

```
CONFIG = {
    "color": BLACK,
    "fill_opacity": 0,
    "stroke_width": 0,
    "artificial_width": 0.01,
    "artificial_height": 0.01,
}
```

Functions

Functions defined in class `VectorizedPoint` are

- def __init__(self, location=ORIGIN, **kwargs)
- def get_width(self)
- def get_height(self)
- def get_location(self)
- def set_location(self, new_loc)

Class CurvesAsSubmobjects(VGroup)

class manimlib.mobject.types.vectorized_mobject.CurvesAsSubmobjects(VGroup) version 19Dec2019

Functions

Functions defined in class `CurvesAsSubmobjects` are

- def __init__(self, vmobject, **kwargs)

Class DashedVMobject(VMobject)

> class manimlib.mobject.types.vectorized_mobject.DashedVMobject(VMobject) version 19Dec2019

Configuration

The configuration is defined in manimlib.mobject.types.vectorized_mobject.py

```
CONFIG = {
    "num_dashes": 15,
    "positive_space_ratio": 0.5,
    "color": WHITE
}
```

Functions

Functions defined in class `DashedVMobject(VMobject)` are

- def __init__(self, vmobject, **kwargs)

Source and Reference

https://github.com/3b1b/manim version 19Dec2019

ID: 200302102 Last Updated: 3/21/2020 Revision: 0

Latest Updated Links

- Windows 8.1 Knowledge Base Networking Network Shell Netsh mbn (last updated On 3/28/2021)
- Windows 8.1 Knowledge Base Networking Network Shell Netsh interface portproxy (last updated On 3/27/2021)
- Windows 8.1 Knowledge Base Networking Network Shell Netsh http (last updated On 3/26/2021)
- Windows 8.1 Knowledge Base Networking Network Shell Netsh (last updated On 3/25/2021)
- Manim Knowledge Base Getting Started Useful Information Geometry VMobject TipableVMobject Line Number_line.py (last updated On 3/24/2021)
- Manim Knowledge Base Getting Started Useful Information Numbers DecimalNumber, Integer (last updated On 3/23/2021)
- Manim Knowledge Base Getting Started Useful Information Numbers (last updated On 3/22/2021)
- Manim Knowledge Base Getting Started Useful Information Geometry VMobject TipableVMobject Line DashedLine, TangentLine, Arrow, Vector, DoubleArrow (last updated On 3/21/2021)
- Manim Knowledge Base Getting Started Useful Information Geometry VMobject TipableVMobject Arc Circle, Dot, SmallDot, Ellipse, Annulus (last updated On 3/20/2021)
- Manim Knowledge Base Getting Started Useful Information Geometry VMobject TipableVMobject Arc ArcBetweenPoints, CurvedArrow, CurvedDoubleArrow (last updated On 3/19/2021)
- Manim Knowledge Base Getting Started Useful Information Geometry VMobject TipableVMobject Line (last updated On 3/18/2021)



Nu Html Checker 53 na na