```
<u>light</u> <u>dark</u>
                                                        Ursina cheat sheet
Entity
<u>Text</u>
                                                        This document lists most modules and classes in ursina. Each
Button
                                               section is structured as follows:
mouse
                                                       ClassName(BaseClass)
<u>raycaster</u>
                                                            module location
<u>application</u>
<u>build</u>
                                                            parameters
                                                                How instantiate the class, ie. Button(text='',
camera
color
                                               **kwargs).
                                                                 '**kwargs' in this case, means you can give it optional
curve
<u>duplicate</u>
                                               keyword arguments.
                                                                 For example, Button('Start', scale=.25,
input handler
                                               color=color.blue, position=(-.1,.25)) also includes
<u>main</u>
                                                                 information on how big the button should be, its color
mesh_importer
<u>scene</u>
                                               and its position.
<u>shader</u>
                                                            attributes
string_utilities
                                                                Names of values we can get/set, sometimes followed by
<u>text</u>
<u>texture_importer</u>
                                               its starting value and a short explanation.
                                                                For example, 'scale', 'color' and 'position' are
<u>ursinamath</u>
ursinastuff
                                                                attributes we gave the Button above. These are members
                                              of Entity, which Button class
<u>window</u>
                                                                 inherits from, so the Button class can also access
<u>Audio</u>
                                              these.
Collider
<u>BoxCollider</u>
                                                            methods/functions
                                                                these ends with (), which means they are functions that
<u>SphereCollider</u>
                                              can be called.
<u>MeshCollider</u>
<u>CollisionZone</u>
                                                                Also lists their parameters and default arguments.
                                                                 For example, Entity has a method called 'look_at()'. You
Color
                                              need to give it a
CubicBezier
<u>HitInfo</u>
                                                                 'target' (an Entity or position) to look at and
                                               optionally say
<u>Keys</u>
                                                                which axis will be facing the target.
<u>Light</u>
DirectionalLight
                                                            example
PointLight
<u>AmbientLight</u>
                                                        You can search the document with Ctrl+F for instant search
<u>SpotLight</u>
<u>Ursina</u>
                                               results.
<u>MeshModes</u>
<u>Mesh</u>
                                               Entity()
<u>Wait</u>
                                                 ursina.entity
<u>Func</u>
<u>Sequence</u>
                                                 Entity(add_to_scene_entities=True, **kwargs)
<u>Shader</u>
<u>Texture</u>
                                                 name = camel_to_snake(self.type)
<u>Trigger</u>
                                                 enabled = True
                                                                     # disabled entities will not be visible nor run
<u>Empty</u>
                                                 code.
LoopingList
                                                 visible = True
<u>Vec2</u>
                                                ignore = False
                                                                    # if True, will not try to run code.
<u>Vec3</u>
                                                 eternal = False  # eternal entities does not get destroyed on
                                                scene.clear()
Animation
                                                 ignore paused = False
                                                                              # if True, will still run when application
Animator
                                                 is paused. useful when making a pause menu for example.
<u>ButtonGroup</u>
                                                 ignore_input = False
ButtonList
                                                 parent = scene
                                                                    # default parent is scene, which means it's in 3d
<u>Node</u>
                                                space. to use UI space, set the parent to camera.ui instead.
<u>Conversation</u>
                                                 add_to_scene_entities = add_to_scene_entities # set to False to be
Cursor
                                                ignored by the engine, but still get rendered.
DebugMenu
                                                                      # set model with model='model name' (without file
                                                model = None
<u>Draggable</u>
                                                type extension)
<u>DropdownMenuButton</u>
                                                 color = color.white
<u>DropdownMenu</u>
                                                texture = None
                                                                     # set model with texture='texture name'. requires a
<u>EditorCamera</u>
                                                model to be set beforehand.
ExitButton
                                                render_queue = 0
FileButton
                                                double_sided = False
<u>FileBrowser</u>
                                                collision = False # toggle collision without changing collider.
<u>FileBrowserSave</u>
                                                                     # set to 'box'/'sphere'/'mesh' for auto fitted
                                                 collider = None
<u>FirstPersonController</u>
                                                 collider.
FrameAnimation3d
                                                 scripts = list() # add with add_script(class_instance). will assign
<u>GridEditor</u>
                                                 an 'entity' variable to the script.
<u>PixelEditor</u>
ASCIIEditor
                                                 animations = list()
                                                 hovered = False
                                                                     # will return True if mouse hovers entity.
<u>HealthBar</u>
```

origin = Vec3(0,0,0)

<u>HotReloader</u>

```
position = Vec3(0,0,0) # right, up, forward. can also set self.x,
                                                 self.y, self.z
                                                 rotation = Vec3(0,0,0) # can also set self.rotation_x,
light dark
                                                 self.rotation_y, self.rotation_z
                                                 scale = Vec3(1,1,1)
                                                                           # can also set self.scale x, self.scale y,
Entity
                                                 self.scale z
                                                 line definition = None # returns a Traceback(filename, lineno,
<u>Text</u>
Button
                                                 function, code_context, index).
mouse
                                                 world_parent
                                                 type
                                                                        # get class name.
<u>raycaster</u>
                                                 types
                                                                        # get all class names including those this
<u>application</u>
                                                 inhertits from.
                                                 visible self
                                                                        # set visibility of self, without affecting
build
camera
                                                 children.
color
                                                 origin x
curve
                                                 origin_y
<u>duplicate</u>
                                                 origin z
                                                 world position
input handler
                                                 world x
<u>main</u>
                                                 world_y
mesh_importer
<u>scene</u>
                                                 world z
shader
                                                 Х
string_utilities
                                                 У
text
                                                 z
<u>texture_importer</u>
                                                 Χ
                                                                        # shortcut for int(entity.x)
                                                 Υ
<u>ursinamath</u>
                                                                        # shortcut for int(entity.y)
<u>ursinastuff</u>
                                                 7
                                                                        # shortcut for int(entity.z)
<u>window</u>
                                                 world rotation
                                                 world_rotation_x
                                                 world_rotation_y
<u>Audio</u>
                                                 world rotation z
Collider
BoxCollider
                                                 rotation_x
<u>SphereCollider</u>
                                                 rotation_y
                                                 rotation_z
<u>MeshCollider</u>
<u>CollisionZone</u>
                                                 quat
                                                 world scale
Color
CubicBezier
                                                 world scale x
HitInfo
                                                 world scale v
                                                 world scale z
<u>Keys</u>
                                                 scale_x
<u>Light</u>
                                                 scale_y
DirectionalLight
PointLight
                                                 scale_z
<u>AmbientLight</u>
                                                 transform
                                                                        # get/set position, rotation and scale
                                                 world_transform
                                                                        # get/set world position, world rotation and
<u>SpotLight</u>
<u>Ursina</u>
                                                 world scale
<u>MeshModes</u>
                                                 forward
                                                                        # get forward direction.
<u>Mesh</u>
                                                 back
                                                                        # get backwards direction.
<u>Wait</u>
                                                 right
                                                                        # get right direction.
<u>Func</u>
                                                 left
                                                                        # get left direction.
                                                                        # get up direction.
<u>Sequence</u>
                                                 up
                                                                        # get down direction.
<u>Shader</u>
                                                 down
                                                                        # get screen position(ui space) from world space.
<u>Texture</u>
                                                 screen position
<u>Trigger</u>
                                                 shader
<u>Empty</u>
                                                 texture_scale
                                                 texture_offset
LoopingList
                                                 tileset size
                                                                        # if the texture is a tileset, say how many tiles
Vec2
                                                 there are so it only use one tile of the texture, e.g. tileset_size=
<u>Vec3</u>
Animation
                                                 tile coordinate
                                                                        # set the tile coordinate, starts in the lower
Animator
                                                 left.
ButtonGroup
                                                 alpha
ButtonList
                                                 always_on_top
                                                 unlit
Node
                                                 billboard
                                                                        # set to True to make this Entity always face the
<u>Conversation</u>
Cursor
                                                 camera.
DebugMenu
                                                 model_bounds
<u>Draggable</u>
                                                 model_center
                                                 bounds
<u>DropdownMenuButton</u>
                                                 children
<u>DropdownMenu</u>
EditorCamera
                                                 attributes
                                                                        # attribute names. used by duplicate().
ExitButton
FileButton
                                                 enable()
                                                 disable()
<u>FileBrowser</u>
                                                 set_shader_input(name, value)
<u>FileBrowserSave</u>
                                                 generate_sphere_map(size=512,
<u>FirstPersonController</u>
                                                 name=f'sphere_map_{len(scene.entities)}')
FrameAnimation3d
                                                 generate_cube_map(size=512, name=f'cube_map_{len(scene.entities)}')
<u>GridEditor</u>
                                                 reparent to(entity)
<u>PixelEditor</u>
ASCIIEditor
                                                 get position(relative to=scene)
<u>HealthBar</u>
                                                 set_position(value, relative_to=scene)
<u>HotReloader</u>
```

```
add_script(class_instance)
                                                 combine(analyze=False, auto_destroy=True, ignore=[])
                                                 flip_faces()
                                                 look_at(target, axis='forward')
light dark
                                                 look_at_2d(target, axis='z')
                                                 has_ancestor(possible_ancestor)
Entity
                                                 animate(name, value, duration=.1, delay=0, curve=curve.in_expo,
<u>Text</u>
Button
                                                 loop=False, resolution=None, interrupt='kill', time_step=None,
                                                 auto_destroy=True)
                                                 animate_position(value, duration=.1, **kwargs)
<u>raycaster</u>
                                                 animate_rotation(value, duration=.1, **kwargs)
                                                 animate_scale(value, duration=.1, **kwargs)
<u>application</u>
                                                 animate_{e}(value, duration=.1, delay=0, **kwargs)
build
                                                 shake(duration=.2, magnitude=1, speed=.05, direction=(1,1))
camera
                                                 animate_color(value, duration=.1, interrupt='finish', **kwargs)
fade_out(value=0, duration=.5, **kwargs)
color
curve
                                                 fade_in(value=1, duration=.5, **kwargs)
<u>duplicate</u>
                                                 blink(value=color.clear, duration=.1, delay=0,
input handler
                                                 curve=curve.in_expo_boomerang, interrupt='finish', **kwargs)
<u>main</u>
                                                 intersects(traverse_target=scene, ignore=(), debug=False)
mesh_importer
<u>scene</u>
                                                   e = Entity(model='quad', color=color.orange, position=(0,0,1),
shader
string_utilities
                                                   scale=1.5, rotation=(0,0,45), texture='brick')
text
<u>texture_importer</u>
                                                   '''example of inheriting Entity'''
<u>ursinamath</u>
                                                   class Player(Entity):
                                                        def __init__(self, **kwargs):
<u>ursinastuff</u>
                                                            super().__init__()
<u>window</u>
                                                            self.model='cube
                                                            self.color = color.red
Audio
                                                            self.scale_y = 2
Collider
BoxCollider
                                                            for key, value in kwargs.items():
<u>SphereCollider</u>
                                                                 setattr(self, key, value)
<u>MeshCollider</u>
<u>CollisionZone</u>
                                                        def input(self, key):
Color
                                                            if key == 'space':
CubicBezier
                                                                 self.animate_x(2, duration=1)
HitInfo
<u>Keys</u>
<u>Light</u>
                                                        def update(self):
                                                            self.x += held_keys['d'] * time.dt * 10
<u>DirectionalLight</u>
                                                            self.x -= held_keys['a'] * time.dt * 10
PointLight
<u>AmbientLight</u>
SpotLight
                                                   player = Player(x=-1)
<u>Ursina</u>
MeshModes
<u>Mesh</u>
<u>Wait</u>
<u>Func</u>
                                               Text(Entity)
<u>Sequence</u>
                                                 ursina.text
Shader
<u>Texture</u>
                                                 Text(text='', start_tag=start_tag, end_tag=end_tag, ignore=True,
<u>Trigger</u>
                                                 **kwargs)
<u>Empty</u>
LoopingList
                                                 size = Text.size
Vec2
                                                 parent = camera.ui
<u>Vec3</u>
                                                 text_nodes = list()
                                                 images = list()
Animation
                                                 origin = (-.5, .5)
Animator
                                                 font = Text.default font
<u>ButtonGroup</u>
                                                 resolution = Text.default_resolution
ButtonList
                                                 line_height = 1
Node
                                                 use_tags = True
<u>Conversation</u>
                                                 start_tag = start_tag
Cursor
                                                 end_tag = end_tag
DebugMenu
                                                 text_colors = {'default' : color.text_color}
<u>Draggable</u>
                                                 tag = Text.start tag+'default'+Text.end tag
<u>DropdownMenuButton</u>
                                                 current_color = self.text_colors['default']
<u>DropdownMenu</u>
                                                 scale override = 1
EditorCamera
                                                 appear_sequence = None # gets created when calling appear()
ExitButton
                                                 text
FileButton
                                                 color
                                                                       # sets the default color.
<u>FileBrowser</u>
                                                 width
                                                                       # gets the width of the widest line.
<u>FileBrowserSave</u>
                                                 height
                                                                       # gets the height of the text
<u>FirstPersonController</u>
FrameAnimation3d
                                                                       # set this to make the text wrap after a certain
                                                 wordwrap
<u>GridEditor</u>
                                                 number of characters.
<u>PixelEditor</u>
                                                 background
ASCIIEditor
<u>HealthBar</u>
                                                 create_text_section(text, tag='', x=0, y=0)
```

<u>HotReloader</u>

```
create_background(padding=size*2, radius=size,
                                                   color=ursina.color.black66)
                                                   appear(speed=.025, delay=0)
light dark
                                                   get_width(string, font=None)
Entity
                                                     from ursina import *
<u>Text</u>
Button
                                                       app = Ursina()
mouse
                                                       descr = dedent(
                                                            <scale:1.5><orange>Rainstorm<default><scale:1>
<u>raycaster</u>
                                                            Summon a <azure>rain storm <default>to deal 5 <azure>water
<u>application</u>
                                                            damage <default>to <hsb(0,1,.7)>everyone, <default>
build
camera
                                                     <image:file_icon> <red><image:file_icon> test <default>including
color
                                                     <orange>yourself. <default>
                                                            Lasts for 4 rounds.''').strip()
curve
<u>duplicate</u>
                                                       Text.default resolution = 1080 * Text.size
input handler
                                                       test = Text(text=descr, origin=(0,0), background=True)
<u>main</u>
mesh_importer
<u>scene</u>
shader
string_utilities
                                                       def input(key):
text
<u>texture_importer</u>
                                                            if key == 'a':
                                                                print('a')
<u>ursinamath</u>
                                                                 test.text = '<default><image:file_icon> <red>
ursinastuff
<u>window</u>
                                                     <image:file icon> test
                                                                print('by', test.text)
<u>Audio</u>
Collider
                                                       window.fps counter.enabled = False
<u>BoxCollider</u>
                                                       print('....', Text.get_width('yolo'))
<u>SphereCollider</u>
                                                       app.run()
<u>MeshCollider</u>
<u>CollisionZone</u>
Color
CubicBezier
HitInfo
                                                 Button(<u>Entity</u>)
<u>Keys</u>
                                                   ursina.prefabs.button
<u>Light</u>
DirectionalLight
                                                   Button(text='', radius=.1, **kwargs)
PointLight
<u>AmbientLight</u>
                                                   parent = camera.ui
<u>SpotLight</u>
                                                   collider = 'box'
<u>Ursina</u>
                                                   disabled = False
<u>MeshModes</u>
                                                   color = Button.color
<u>Mesh</u>
                                                   text entity = None
<u>Wait</u>
                                                   highlight_color = self.color.tint(.2)
<u>Func</u>
                                                   pressed_color = self.color.tint(-.2)
<u>Sequence</u>
                                                   highlight_scale = 1  # multiplie
pressed_scale = 1  # multiplier
                                                                            # multiplier
<u>Shader</u>
<u>Texture</u>
                                                   original scale = self.scale
<u>Trigger</u>
                                                   icon = None
<u>Empty</u>
                                                   text
LoopingList
                                                   text_origin
Vec2
                                                   text_color
<u>Vec3</u>
                                                   input(key)
Animation
                                                   on_mouse_enter()
Animator
                                                   on mouse exit()
<u>ButtonGroup</u>
                                                   on_click()
ButtonList
                                                   fit_to_text(radius=.1)
<u>Node</u>
<u>Conversation</u>
                                                     b = Button(text='hello world!', color=color.azure, icon='sword',
Cursor
                                                     scale=.25, text_origin=(-.5,0))
DebugMenu
                                                     b.on_click = application.quit # assign a function to the button.
Draggable
                                                   b.tooltip = Tooltip('exit')
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
EditorCamera
ExitButton
FileButton
                                                mouse
<u>FileBrowser</u>
<u>FileBrowserSave</u>
                                                   <u>ursina.mouse</u>
<u>FirstPersonController</u>
FrameAnimation3d
                                                   enabled = False
<u>GridEditor</u>
                                                   visible = True
```

locked = False

position = Vec3(0,0,0)

delta = Vec3(0,0,0)

<u>PixelEditor</u>

**ASCIIEditor** 

<u>HotReloader</u>

<u>HealthBar</u>

align()

```
prev_x = 0
                                                 prev_y = 0
                                                 start_x = 0
<u>light</u> <u>dark</u>
                                                 start y = 0
                                                 velocity = Vec3(0,0,0)
Entity
                                                 prev_click_time = time.time()
<u>Text</u>
                                                 prev_click_pos = None
Button
                                                 double_click_distance = .5
                                                 double_click_movement_limit = .01
mouse
<u>raycaster</u>
                                                 hovered_entity = None # returns the closest hovered entity with a
                                                 collider.
<u>application</u>
                                                 left = False
                                                 right = False
build
camera
                                                 middle = False
color
                                                 delta drag = Vec3(0,0,0)
                                                 update\_step = 1
curve
<u>duplicate</u>
                                                 traverse target = scene # set this to None to disable collision with
                                                 scene, which might be a good idea if you have lots of colliders.
input handler
                                                 raycast = True
<u>main</u>
                                                 collision = None
mesh_importer
<u>scene</u>
                                                 collisions = list()
shader
                                                 enabled = True
string_utilities
text
<u>texture_importer</u>
                                                 normal
                                                                        # returns the normal of the polygon, in local
<u>ursinamath</u>
                                                 space.
<u>ursinastuff</u>
                                                 world_normal
                                                                        # returns the normal of the polygon, in world
<u>window</u>
                                                 space.
                                                 point
                                                                        # returns the point hit, in local space
                                                                        # returns the point hit, in world space
<u>Audio</u>
                                                 world_point
Collider
<u>BoxCollider</u>
                                                 input(key)
<u>SphereCollider</u>
                                                 update()
<u>MeshCollider</u>
                                                 find_collision()
<u>CollisionZone</u>
                                                 unhover_everything_not_hit()
Color
CubicBezier
                                                    Button(parent=scene, text='a')
<u>HitInfo</u>
                                                    def update():
<u>Keys</u>
<u>Light</u>
                                                        print(mouse.position, mouse.point)
DirectionalLight
PointLight
                                                    Cursor()
                                                   mouse.visible = False
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
<u>Mesh</u>
<u>Wait</u>
                                                raycaster
<u>Func</u>
                                                 ursina.raycaster
<u>Sequence</u>
<u>Shader</u>
<u>Texture</u>
                                                 distance(a, b)
<u>Trigger</u>
                                                 raycast(origin, direction=(0,0,1), distance=inf,
<u>Empty</u>
                                                 traverse_target=scene, ignore=list(), debug=False)
LoopingList
                                                 boxcast(origin, direction=(0,0,1), distance=9999, thickness=(1,1),
Vec2
                                                 traverse_target=scene, ignore=list(), debug=False) # similar to
<u>Vec3</u>
                                                 raycast, but with width and height
Animation
Animator
                                                   Casts a ray from *origin*, in *direction*, with length *distance* and
<u>ButtonGroup</u>
ButtonList
                                                   a HitInfo containing information about what it hit. This ray will
Node
                                                   only hit entities with a collider.
<u>Conversation</u>
Cursor
                                                   Use optional *traverse_target* to only be able to hit a specific
DebugMenu
                                                   entity and its children/descendants.
<u>Draggable</u>
                                                   Use optional *ignore* list to ignore certain entities.
<u>DropdownMenuButton</u>
                                                   Setting debug to True will draw the line on screen.
DropdownMenu
<u>EditorCamera</u>
                                                    Example where we only move if a wall is not hit:
ExitButton
FileButton
<u>FileBrowser</u>
<u>FileBrowserSave</u>
                                                   class Player(Entity):
<u>FirstPersonController</u>
FrameAnimation3d
                                                        def update(self):
<u>GridEditor</u>
                                                             self.direction = Vec3(
<u>PixelEditor</u>
                                                                  self.forward * (held_keys['w'] - held_keys['s'])
ASCIIEditor
                                                                 + self.right * (held_keys['d'] - held_keys['a'])
HealthBar
                                                                 ).normalized() # get the direction we're trying to walk
<u>HotReloader</u>
```

```
light dark
Entity
<u>Text</u>
Button
mouse
<u>raycaster</u>
<u>application</u>
build
camera
color
curve
<u>duplicate</u>
input handler
<u>main</u>
mesh_importer
<u>scene</u>
<u>shader</u>
string_utilities
text
<u>texture_importer</u>
<u>ursinamath</u>
ursinastuff
<u>window</u>
<u>Audio</u>
Collider
<u>BoxCollider</u>
<u>SphereCollider</u>
<u>MeshCollider</u>
<u>CollisionZone</u>
Color
CubicBezier
HitInfo
<u>Keys</u>
<u>Light</u>
DirectionalLight
PointLight
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
<u>Mesh</u>
<u>Wait</u>
<u>Func</u>
<u>Sequence</u>
<u>Shader</u>
<u>Texture</u>
<u>Trigger</u>
<u>Empty</u>
LoopingList
Vec2
<u>Vec3</u>
Animation
Animator
ButtonGroup
ButtonList
Node
<u>Conversation</u>
Cursor
DebugMenu
<u>Draggable</u>
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
EditorCamera
ExitButton
FileButton
<u>FileBrowser</u>
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
FrameAnimation3d
<u>GridEditor</u>
<u>PixelEditor</u>
ASCIIEditor
<u>HealthBar</u>
<u>HotReloader</u>
```

## application

ursina.application

```
paused = False
time_scale = 1
sequences = list()
trace entity definition = False # enable to set entity.line definition
package_folder = Path(__file__).parent
blender_paths = dict()
development mode = True
scenes_folder = asset_folder / 'scenes/'
scripts_folder = asset_folder / 'scripts/'
fonts_folder = asset_folder / 'fonts/'
compressed_textures_folder = asset_folder / 'textures_compressed/'
compressed_models_folder = asset_folder / 'models_compressed/'
base = None
                         # this will be set once the Ursina() is
hot_reloader = None  # will be set my main if development_mode
pause()
resume()
quit()
load_settings(path=asset_folder / 'settings.py')
```

## build

<u>ursina.</u>init\_

python -m ursina.build

open cmd at your project folder and run 'python -m ursina.build' to package your app for windows.

#### camera

```
parent = scene
name = 'camera'
eternal = True
ui_size = 40
ui = None
fov = 40
orthographic = False
clip_plane_near
clip_plane_far
aspect_ratio
shader

set_up()
set_shader_input(name, value)
```

```
light dark
Entity
<u>Text</u>
Button
mouse
<u>raycaster</u>
<u>application</u>
build
camera
color
curve
<u>duplicate</u>
input handler
<u>main</u>
mesh_importer
<u>scene</u>
shader
string_utilities
text
<u>texture_importer</u>
<u>ursinamath</u>
<u>ursinastuff</u>
<u>window</u>
<u>Audio</u>
Collider
BoxCollider
<u>SphereCollider</u>
<u>MeshCollider</u>
CollisionZone
Color
CubicBezier
HitInfo
<u>Keys</u>
<u>Light</u>
DirectionalLight
PointLight
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
<u>Mesh</u>
Wait
<u>Func</u>
<u>Sequence</u>
<u>Shader</u>
<u>Texture</u>
<u>Trigger</u>
<u>Empty</u>
LoopingList
Vec2
Vec3
Animation
Animator
ButtonGroup
ButtonList
Node
<u>Conversation</u>
Cursor
DebugMenu
<u>Draggable</u>
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
EditorCamera
ExitButton
FileButton
<u>FileBrowser</u>
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
FrameAnimation3d
<u>GridEditor</u>
PixelEditor
ASCIIEditor
HealthBar
<u>HotReloader</u>
```

```
window.borderless = False
camera.orthographic = True
e = Entity()
e.model = 'quad'
e.color = color.random_color()
e.position = (-2, 0, 10)
e = Entity()
e.model = 'quad'
e.color = color.random_color()
e.position = (2, 0, 10)
e = Entity()
e.model = 'quad'
e.color = color.random color()
e.position = (0, 0, 40)
EditorCamera()
from ursina.shaders import camera grayscale shader
camera.shader = camera_grayscale_shader
```

## color

ursina.color

```
color = hsv
  white = color(0, 0, 1)

smoke = color(0, 0, 0.96)
  light_gray = color(0, 0, 0.75)
 gray = color(0, 0, 0.5)

dark\_gray = color(0, 0, 0.25)

black = color(0, 0, 0)

red = color(0, 1, 1)
black = color(0, 0, 0)

red = color(0, 1, 1)

yellow = color(60, 1, 1)

lime = color(90, 1, 1)

green = color(120, 1, 1)

turquoise = color(150, 1, 1)

cyan = color(210, 1, 1)

blue = color(240, 1, 1)

violet = color(270, 1, 1)

magenta = color(300, 1, 1)

pink = color(300, 1, 1)

pink = color(330, 1, 1)

brown = rgb(165, 42, 42)

olive = rgb(128, 128, 0)

peach = rgb(255, 218, 185)

gold = rgb(255, 215, 0)

salmon = rgb(250, 128, 114)

clear = color(0, 0, 0, 0, 0)

white10 = color(1,1,1, 0.10)

white33 = color(1,1,1, 0.50)

white66 = color(1,1,1, 0.66)

black10 = color(0,0,0, 0.33)

black50 = color(0,0,0, 0.50)

black66 = color(0,0,0, 0.66)
  black66 =
                                  Color(0,0,0, 0.66)
  black90 =
                                   Color(0,0,0, 0.90)
  text = smoke
  light_text = smoke
  dark_text = color(0, 0, .1)
  text color = light text
  color_names = ('white', 'smoke', 'light_gray', 'gray', 'dark_gray',
  'black',
  colors = dict()
  hsv(h, s, v, a=1)
  rgba(r, g, b, a=255)
  rgb(r, g, b, a=255)
  to_hsv(color)
  hex(value)
  brightness(color)
  inverse(color)
  random_color()
  tint(color, amount=.2)
```

```
print(_3)
<u>light</u> <u>dark</u>
                                                         p = Entity(x=-2)
                                                         for key in color.colors:
Entity
                                                              print(key)
                                                              b = Button(parent=p, model=Quad(subdivisions=2),
<u>Text</u>
                                                         color=color.colors[key], text=key)
b.text_entity.scale *= .5
Button
mouse
<u>raycaster</u>
                                                         grid_layout(p.children, max_x=8)
<u>application</u>
                                                         for name in ('r', 'g', 'b', 'h', 's', 'v', 'brightness'):
    print(name + ':', getattr(color.random_color(), name))
build
camera
color
                                                         e = Entity(model='cube', color=color.lime)
curve
<u>duplicate</u>
                                                        print(e.color.name)
input handler
<u>main</u>
mesh_importer
<u>scene</u>
<u>shader</u>
                                                    curve
string_utilities
                                                      ursina.curve
<u>text</u>
<u>texture_importer</u>
<u>ursinamath</u>
                                                      linear(t)
<u>ursinastuff</u>
                                                      in_sine(t)
<u>window</u>
                                                      out_sine(t)
                                                      in_out_sine(t)
<u>Audio</u>
                                                      in quad(t)
Collider
                                                      out quad(t)
<u>BoxCollider</u>
                                                      in_out_quad(t)
<u>SphereCollider</u>
                                                      in_cubic(t)
<u>MeshCollider</u>
                                                      out_cubic(t)
<u>CollisionZone</u>
                                                      in_out_cubic(t)
Color
                                                      in_quart(t)
CubicBezier
                                                      out_quart(t)
<u>HitInfo</u>
                                                      in_out_quart(t)
<u>Keys</u>
                                                      in_quint(t)
<u>Light</u>
                                                      out_quint(t)
<u>DirectionalLight</u>
                                                      in_out_quint(t)
PointLight
                                                      in_expo(t)
<u>AmbientLight</u>
                                                      out_expo(t)
<u>SpotLight</u>
                                                      in_out_expo(t)
<u>Ursina</u>
                                                      in_circ(t)
<u>MeshModes</u>
                                                      out circ(t)
<u>Mesh</u>
                                                      in out circ(t)
<u>Wait</u>
                                                      in_back(t, magnitude=1.70158)
<u>Func</u>
                                                      out_back(t, magnitude=1.70158)
<u>Sequence</u>
                                                      in_out_back(t, magnitude=1.70158)
Shader
                                                      in_elastic(t, magnitude=.7)
out_elastic(t, magnitude=.7)
<u>Texture</u>
<u>Trigger</u>
                                                      in_out_elastic(t, magnitude=0.65)
<u>Empty</u>
                                                      out_bounce(t)
LoopingList
                                                      in_bounce(t)
Vec2
                                                      in_out_bounce(t)
<u>Vec3</u>
                                                      {e}_boomerang(t)
Animation
                                                       '''Draws a sheet with every curve and its name'''
Animator
<u>ButtonGroup</u>
                                                         camera.orthographic = True
ButtonList
                                                         camera.fov = 16
<u>Node</u>
                                                         camera.position = (9, 6)
<u>Conversation</u>
                                                         window.color = color.black
Cursor
DebugMenu
                                                         i = 0
Draggable
                                                         for e in dir(curve):
<u>DropdownMenuButton</u>
                                                              try:
<u>DropdownMenu</u>
                                                                   item = getattr(curve, e)
<u>EditorCamera</u>
                                                                   print(item.__name__, ':', item(.75))
<u>ExitButton</u>
                                                                   curve renderer = Entity(
FileButton
                                                                       model=Mesh(vertices=[Vec3(i / 31, item(i / 31), 0) for i
<u>FileBrowser</u>
                                                         in range(32)], mode='line', thickness=2),
<u>FileBrowserSave</u>
                                                                       color=color.light_gray)
<u>FirstPersonController</u>
                                                                   row = floor(i / 8)
FrameAnimation3d
                                                                   curve_renderer.x = (i \% 8) * 2.5
<u>GridEditor</u>
                                                                   curve_renderer.y = row * 1.75
<u>PixelEditor</u>
                                                                   label = Text(parent=curve_renderer, text=item.__name__,
ASCIIEditor
                                                         scale=8, color=color.gray, y=-.1)
<u>HealthBar</u>
                                                                   i += 1
<u>HotReloader</u>
```

print(color.brightness(color.blue))

```
c = CubicBezier(0, .5, 1, .5)
<u>light</u> <u>dark</u>
                                                      print('----', c.calculate(.23))
Entity
<u>Text</u>
                                                      window.exit_button.visible = False
Button
                                                      window.fps_counter.enabled = False
mouse
                                                      These are used by Entity when animating, like this:
<u>raycaster</u>
<u>application</u>
                                                      e = Entity()
<u>build</u>
                                                      e.animate_y(1, curve=curve.in_expo)
camera
color
                                                      e2 = Entity(x=1.5)
                                                      e2.animate_y(1, curve=curve.CubicBezier(0,.7,1,.3))
curve
<u>duplicate</u>
input handler
<u>main</u>
mesh_importer
<u>scene</u>
<u>shader</u>
                                                  duplicate
string_utilities
                                                    ursina.duplicate
<u>text</u>
<u>texture_importer</u>
<u>ursinamath</u>
                                                    duplicate(entity, copy_children=True, **kwargs): # use a for loop
ursinastuff
                                                    instead of duplicate() # use a for loop instead of duplicate() if
<u>window</u>
                                                   you can.
<u>Audio</u>
                                                      e = Button(parent=scene, scale=1, text='yolo')
Collider
                                                      e2 = duplicate(e, x=1.25)
<u>BoxCollider</u>
                                                    EditorCamera()
<u>SphereCollider</u>
<u>MeshCollider</u>
<u>CollisionZone</u>
Color
CubicBezier
                                                  input handler
<u>HitInfo</u>
<u>Keys</u>
                                                    ursina.input handler
<u>Light</u>
DirectionalLight
                                                    held keys = defaultdict(lambda: 0)
PointLight
                                                    rebinds = dict()
<u>AmbientLight</u>
<u>SpotLight</u>
                                                    bind(original_key, alternative_key)
<u>Ursina</u>
                                                    unbind(key)
<u>MeshModes</u>
                                                    rebind(to_key, from_key)
<u>Mesh</u>
                                                    input(key)
<u>Wait</u>
<u>Func</u>
                                                      input_handler.bind('s', 'arrow down') # 's'-key will now be
<u>Sequence</u>
                                                      registered as 'arrow down'-key
<u>Shader</u>
<u>Texture</u>
                                                      def test():
<u>Trigger</u>
                                                           print('----')
<u>Empty</u>
                                                      def input(key):
LoopingList
                                                           print(key)
Vec2
                                                           if key == 'left mouse down':
<u>Vec3</u>
                                                               print('pressed left mouse button')
Animation
                                                           if key == Keys.left_mouse_down: # same as above, but with Keys
Animator
ButtonGroup
                                                               print('pressed left mouse button')
ButtonList
<u>Node</u>
<u>Conversation</u>
                                                      def update():
Cursor
                                                           for key, value in held_keys.items():
DebugMenu
                                                               if value != 0:
<u>Draggable</u>
                                                                   print(key, value)
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
<u>EditorCamera</u>
ExitButton
FileButton
                                                 main
<u>FileBrowser</u>
                                                    ursina.main
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
                                                    time.dt = 0
FrameAnimation3d
<u>GridEditor</u>
<u>PixelEditor</u>
                                                      app = Ursina()
ASCIIEditor
                                                      app.run()
<u>HealthBar</u>
<u>HotReloader</u>
```

except:

```
light dark
                                                   mesh importer
Entity
                                                      ursina.mesh importer
<u>Text</u>
Button
                                                      blender scenes = dict()
mouse
<u>raycaster</u>
                                                      load_model(name, path=application.asset_folder, file_types=('.bam',
'.ursinamesh', '.obj', '.glb', '.gltf', '.blend'), use_deepcopy=False)
load_blender_scene(name, path=application.asset_folder, load=True,
<u>application</u>
build
                                                      reload=False, skip_hidden=True, models_only=False)
camera
                                                      get_blender(blend_file) # try to get a matching blender version in
color
                                                      case we have multiple blender version installed
curve
                                                      compress_models(path=None,
<u>duplicate</u>
                                                      outpath=application.compressed_models_folder, name='*')
input handler
                                                      obj_to_ursinamesh()
<u>main</u>
                                                      compress_models_fast(model_name=None, write_to_disk=False)
mesh_importer
                                                      ursina_mesh_to_obj(mesh, name='',
<u>scene</u>
                                                      out_path=application.compressed_models_folder, max_decimals=3)
<u>shader</u>
                                                      compress_internal()
string_utilities
<u>text</u>
                                                        print('imported_meshes:\n', imported_meshes)
<u>texture_importer</u>
<u>ursinamath</u>
                                                        t = time.time()
ursinastuff
<u>window</u>
                                                        application.asset_folder = application.asset_folder.parent /
                                                         'samples'
<u>Audio</u>
                                                        Entity(model='race')
Collider
                                                        Entity(model='ambulance', x=1.5)
<u>BoxCollider</u>
<u>SphereCollider</u>
<u>MeshCollider</u>
<u>CollisionZone</u>
                                                        EditorCamera()
Color
                                                        Sky(texture='sky_sunset')
CubicBezier
<u>HitInfo</u>
<u>Keys</u>
<u>Light</u>
DirectionalLight
                                                    scene
PointLight
<u>AmbientLight</u>
                                                      ursina.scene
<u>SpotLight</u>
<u>Ursina</u>
                                                      render = None
<u>MeshModes</u>
                                                      world = None
<u>Mesh</u>
                                                      camera = None
<u>Wait</u>
                                                      ui camera = None
<u>Func</u>
                                                      entities = []
<u>Sequence</u>
                                                      hidden = NodePath('hidden')
<u>Shader</u>
                                                      reflection_map_name = 'reflection_map_3'
<u>Texture</u>
                                                      fog_color
<u>Trigger</u>
                                                      fog_density
<u>Empty</u>
LoopingList
                                                      set_up()
Vec2
                                                      clear()
<u>Vec3</u>
                                                        e = Entity(model='plane', color=color.black, scale=100)
Animation
                                                        EditorCamera()
Animator
                                                        s = Sky()
ButtonGroup
ButtonList
                                                        def input(key):
<u>Node</u>
                                                             if key == '1':
<u>Conversation</u>
                                                                  for e in scene.entities:
Cursor
                                                                      print(e.name)
DebugMenu
<u>Draggable</u>
                                                             if key == 'd':
<u>DropdownMenuButton</u>
                                                                  scene.clear()
<u>DropdownMenu</u>
<u>EditorCamera</u>
                                                        scene.fog_density = .1
                                                                                               # sets exponential density
<u>ExitButton</u>
                                                        scene.fog_density = (50, 200) # sets linear density start and end
FileButton
<u>FileBrowser</u>
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
FrameAnimation3d
                                                    shader
<u>GridEditor</u>
                                                      ursina.shader
<u>PixelEditor</u>
ASCIIEditor
```

<u>HealthBar</u> <u>HotReloader</u> uniform mat4 p3d\_ModelViewProjectionMatrix;

```
out vec2 uv;
                                                   void main() {
light dark
                                                   uniform sampler2D tex;
                                                   out vec4 color;
Entity
                                                   void main() {
<u>Text</u>
                                                   }
Button
mouse
                                                     from time import perf_counter
<u>raycaster</u>
                                                     t = perf counter()
                                                     Entity(model='cube', shader=Shader())
<u>application</u>
<u>build</u>
                                                     EditorCamera()
camera
                                                     print('tttttttttttt', perf_counter() - t)
color
                                                     def input(key):
                                                          if held_keys['control'] and key == 'r':
curve
<u>duplicate</u>
                                                              reload shaders()
input handler
                                                     def reload_shaders():
<u>main</u>
                                                          for e in scene.entities:
mesh_importer
                                                              if hasattr(e, '_shader'):
    print('-----', e.shader)
<u>scene</u>
shader
string_utilities
text
<u>texture_importer</u>
<u>ursinamath</u>
ursinastuff
                                                 string utilities
<u>window</u>
                                                   ursina.string_utilities
<u>Audio</u>
Collider
                                                   camel to snake(value)
<u>BoxCollider</u>
                                                   snake_to_camel(value)
<u>SphereCollider</u>
                                                   multireplace(string, replacements, ignore_case=False)
<u>MeshCollider</u>
                                                   printvar(var)
<u>CollisionZone</u>
                                                   print_info(str, *args)
Color
                                                   print_warning(str, *args)
<u>CubicBezier</u>
HitInfo
                                                     print(camel_to_snake('CamelToSnake'))
<u>Keys</u>
                                                     print(snake_to_camel('snake_to_camel'))
<u>Light</u>
                                                    printvar('test')
DirectionalLight
PointLight
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
                                                 text
<u>MeshModes</u>
<u>Mesh</u>
                                                   ursina.text
<u>Wait</u>
<u>Func</u>
<u>Sequence</u>
                                                   get_width(string, font=None)
<u>Shader</u>
<u>Texture</u>
                                                     from ursina import *
<u>Trigger</u>
                                                       app = Ursina()
<u>Empty</u>
                                                        descr = dedent('''
LoopingList
                                                            <scale:1.5><orange>Rainstorm<default><scale:1>
Vec2
                                                            Summon a <azure>rain storm <default>to deal 5 <azure>water
<u>Vec3</u>
                                                            damage <default>to <hsb(0,1,.7)>everyone, <default>
Animation
                                                     <image:file_icon> <red><image:file_icon> test <default>including
Animator
                                                     <orange>yourself. <default>
ButtonGroup
                                                            Lasts for 4 rounds.''').strip()
ButtonList
Node
                                                       Text.default resolution = 1080 * Text.size
<u>Conversation</u>
                                                       test = Text(text=descr, origin=(0,0), background=True)
Cursor
DebugMenu
Draggable
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
                                                       def input(key):
<u>EditorCamera</u>
                                                            if key == 'a':
ExitButton
                                                                 print('a')
FileButton
                                                                 test.text = '<default><image:file_icon> <red>
<u>FileBrowser</u>
                                                     <image:file icon> test
<u>FileBrowserSave</u>
                                                                 print('by', test.text)
<u>FirstPersonController</u>
FrameAnimation3d
                                                       window.fps_counter.enabled = False
<u>GridEditor</u>
                                                        print('....', Text.get_width('yolo'))
<u>PixelEditor</u>
                                                        app.run()
ASCIIEditor
<u>HealthBar</u>
```

<u>HotReloader</u>

```
light dark
Entity
<u>Text</u>
Button
mouse
<u>raycaster</u>
<u>application</u>
<u>build</u>
camera
color
curve
<u>duplicate</u>
input handler
                                                              ursinamath
mesh_importer
<u>scene</u>
shader
string_utilities
text
<u>texture_importer</u>
<u>ursinamath</u>
ursinastuff
<u>window</u>
<u>Audio</u>
Collider
BoxCollider
<u>SphereCollider</u>
<u>MeshCollider</u>
<u>CollisionZone</u>
Color
<u>CubicBezier</u>
HitInfo
<u>Keys</u>
<u>Light</u>
DirectionalLight
PointLight
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
<u>Mesh</u>
Wait
<u>Func</u>
<u>Sequence</u>
<u>Shader</u>
<u>Texture</u>
<u>Trigger</u>
<u>Empty</u>
<u>LoopingList</u>
Vec2
Vec3
Animation
Animator
ButtonGroup
ButtonList
Node
<u>Conversation</u>
Cursor
DebugMenu
<u>Draggable</u>
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
EditorCamera
ExitButton
FileButton
<u>FileBrowser</u>
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
FrameAnimation3d
<u>GridEditor</u>
<u>PixelEditor</u>
ASCIIEditor
```

<u>HealthBar</u> <u>HotReloader</u>

```
texture importer
```

```
ursina.texture_importer
 file_types = ('.tif', '.jpg', '.jpeg', '.png', '.gif')
 textureless = False
 load_texture(name, path=None)
 compress_textures(name='')
 Entity(model='quad', texture='white_cube')
ursinamath
```

```
distance(a, b)
distance_2d(a, b)
distance_xz(a, b)
lerp(a, b, t)
inverselerp(a, b, t)
slerp(q1, q2, t)
clamp(value, floor, ceiling)
round_to_closest(value, step=0)
chunk_list(1, chunk_size)
size list()
sum(1)
 e1 = Entity(position = (0,0,0))
 e^2 = Entity(position = (0,1,1))
 distance(e1, e2)
 distance_xz(e1, e2.position)
 between_color = lerp(color.lime, color.magenta, .5)
 print(between_color)
 print(lerp((0,0), (0,1), .5))
 print(lerp(Vec2(0,0), Vec2(0,1), .5))
 print(lerp([0,0], [0,1], .5))
print(round(Vec3(.38, .1351, 353.26), 2))
```

#### ursinastuff

ursinastuff

```
invoke(function, *args, **kwargs)
destroy(entity, delay=0)
find_sequence(name, file_types, folders)
                                                # find frame 0, frame 1,
frame 2 and so on
import_all_classes(path=application.asset_folder, debug=False)
print_on_screen(text, position=window.top_left, origin=(-.5,.5),
scale=1, duration=1)
 def test func(item, x=None, y=None):
      print(item, x, y)
  test_func('test')
 invoke(test_func, 'test', delay=.1)
invoke(test_func, 'test1', 1, 2, delay=.2)
invoke(test_func, 'test2', x=1, y=2, delay=.3)
  def input(key):
      if key == 'space':
           print_on_screen('debug message', position=(0,0), origin=
  (0,0), scale=2)
```

### window

```
# can't be set during play
                                                   vsvnc = True
light dark
                                                   show ursina splash = False
                                                   title = application.asset_folder.name
Entity
                                                   fullscreen_size = Vec2(*self.screen_resolution)
                                                   windowed_size = self.fullscreen_size / 1.25
<u>Text</u>
Button
                                                   windowed_position = None # gets set when entering fullscreen so
                                                   position will be correct when going back to windowed mode
mouse
                                                   forced_aspect_ratio = None # example: window.forced_aspect_ratio
<u>raycaster</u>
                                                   size = self.windowed size
<u>application</u>
                                                   borderless = True
                                                   top = Vec2(0, .5)
build
                                                   bottom = Vec2(0, -.5)
camera
color
                                                   center = Vec2(0, 0)
                                                   color = color.dark_gray
curve
                                                   render_modes = ('default', 'wireframe', 'colliders', 'normals')
<u>duplicate</u>
                                                   render mode = 'default'
input handler
                                                   editor ui = None
<u>main</u>
                                                   left
mesh_importer
<u>scene</u>
                                                   right
<u>shader</u>
                                                   top_left
string_utilities
                                                   top_right
                                                   bottom_left
text
<u>texture_importer</u>
                                                   bottom_right
<u>ursinamath</u>
                                                   position
ursinastuff
                                                   icon
<u>window</u>
                                                   late_init()
<u>Audio</u>
                                                   center_on_screen()
Collider
                                                   make_editor_gui()
                                                                          # called by main after setting up camera and
BoxCollider
                                                   application.development_mode
<u>SphereCollider</u>
                                                   update_aspect_ratio()
<u>MeshCollider</u>
                                                   next_render_mode()
<u>CollisionZone</u>
                                                     app = Ursina(vsync=True)
Color
CubicBezier
                                                     window.title = 'ursina'
HitInfo
<u>Keys</u>
<u>Light</u>
                                                     camera.orthographic = True
                                                     camera.fov = 2
<u>DirectionalLight</u>
PointLight
<u>AmbientLight</u>
SpotLight
<u>Ursina</u>
<u>MeshModes</u>
                                                Audio(Entity)
<u>Mesh</u>
                                                   ursina.audio
<u>Wait</u>
<u>Func</u>
                                                   Audio(sound_file_name='', autoplay=True, auto_destroy=False, **kwargs)
<u>Sequence</u>
Shader
                                                   volume = 1
<u>Texture</u>
                                                   pitch = 1
<u>Trigger</u>
                                                   balance = 0
<u>Empty</u>
                                                   loop = False
LoopingList
                                                   loops = 1
Vec2
                                                   autoplay = autoplay
<u>Vec3</u>
                                                   auto_destroy = auto_destroy
                                                   clip
Animation
                                                   length
Animator
                                                   status
<u>ButtonGroup</u>
                                                   ready
ButtonList
                                                   playing
<u>Node</u>
                                                   time
<u>Conversation</u>
Cursor
                                                   play(start=0)
DebugMenu
                                                   pause()
<u>Draggable</u>
                                                   resume()
<u>DropdownMenuButton</u>
                                                   stop(destroy=True)
<u>DropdownMenu</u>
                                                   fade(value, duration=.5, delay=0, curve=curve.in_expo,
<u>EditorCamera</u>
                                                   resolution=None, interrupt=True)
ExitButton
                                                   fade_in(value=1, duration=.5, delay=0, curve=curve.in_expo,
resolution=None, interrupt='finish',)
FileButton
<u>FileBrowser</u>
                                                   fade_out(value=0, duration=.5, delay=0, curve=curve.in_expo,
<u>FileBrowserSave</u>
                                                   resolution=None, interrupt='finish',)
<u>FirstPersonController</u>
FrameAnimation3d
                                                     from ursina import Ursina, printvar
<u>GridEditor</u>
<u>PixelEditor</u>
                                                     a = Audio('life_is_currency', pitch=1, loop=True, autoplay=True)
ASCIIEditor
                                                     print(a.clip)
<u>HealthBar</u>
```

a.volume=0

<u>HotReloader</u>

ursina.window

```
b = Audio(a.clip)
                                                  def input(key):
                                                       if key == 'f':
light dark
                                                           a.fade_out(duration=4, curve=curve.linear)
Entity
<u>Text</u>
                                                  def update():
Button
                                                      print(a.time)
mouse
<u>raycaster</u>
<u>application</u>
build
                                              Collider()
camera
                                                ursina.collider
color
curve
                                                Collider()
<u>duplicate</u>
input handler
                                                visible
<u>main</u>
mesh_importer
                                                show()
<u>scene</u>
                                                hide()
shader
                                                remove()
string_utilities
text
                                                  e = Entity(model='sphere', x=2)
<u>texture_importer</u>
                                                  e.collider = 'box' # add BoxCollider based on entity's bounds.
<u>ursinamath</u>
                                                  e.collider = 'sphere'
                                                                             # add SphereCollider based on entity's
ursinastuff
                                                  bounds.
<u>window</u>
                                                  e.collider = 'mesh'
                                                                             # add MeshCollider based on entity's bounds.
<u>Audio</u>
                                                  e.collider = BoxCollider(e, center=Vec3(0,0,0), size=Vec3(1,1,1))
Collider
                                                  add BoxCollider at custom positions and size
<u>BoxCollider</u>
                                                  e.collider = SphereCollider(e, center=Vec3(0,0,0), radius=.75)
<u>SphereCollider</u>
                                                  add SphereCollider at custom positions and size
<u>MeshCollider</u>
                                                  e.collider = MeshCollider(e, mesh=e.model, center=Vec3(0,0,0))
<u>CollisionZone</u>
                                                  add MeshCollider with custom shape and center.
Color
CubicBezier
                                                  m = Prismatoid(base_shape=Circle(6), thicknesses=(1, .5))
HitInfo
                                                   e = Button(parent=scene, model=m, collider='mesh', color=color.red,
<u>Keys</u>
                                                  highlight_color=color.yellow)
<u>Light</u>
<u>DirectionalLight</u>
                                                  EditorCamera()
PointLight
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
                                              BoxCollider(Collider)
<u>Mesh</u>
<u>Wait</u>
                                                ursina.collider
<u>Func</u>
<u>Sequence</u>
                                                BoxCollider(entity, center=(0,0,0), size=(1,1,1))
<u>Shader</u>
<u>Texture</u>
                                                shape = CollisionBox(Vec3(center[0], center[1], center[2]), size[0],
<u>Trigger</u>
                                                 size[1], size[2])
<u>Empty</u>
                                                collision_node = CollisionNode('CollisionNode')
LoopingList
                                                node_path = entity.attachNewNode(self.collision_node)
Vec2
                                                visible = False
Vec3
Animation
                                                  e = Entity(model='sphere', x=2)
Animator
                                                  e.collider = 'box'
                                                                             # add BoxCollider based on entity's bounds.
<u>ButtonGroup</u>
                                                  e.collider = 'sphere'
                                                                             # add SphereCollider based on entity's
ButtonList
                                                  bounds.
Node
                                                  e.collider = 'mesh'
                                                                             # add MeshCollider based on entity's bounds.
<u>Conversation</u>
Cursor
                                                  e.collider = BoxCollider(e, center=Vec3(0,0,0), size=Vec3(1,1,1))
DebugMenu
                                                   add BoxCollider at custom positions and size.
<u>Draggable</u>
                                                  e.collider = SphereCollider(e, center=Vec3(0,0,0), radius=.75)
<u>DropdownMenuButton</u>
                                                  add SphereCollider at custom positions and size
<u>DropdownMenu</u>
                                                  e.collider = MeshCollider(e, mesh=e.model, center=Vec3(0,0,0))
<u>EditorCamera</u>
                                                  add MeshCollider with custom shape and center.
ExitButton
FileButton
                                                  m = Prismatoid(base_shape=Circle(6), thicknesses=(1, .5))
<u>FileBrowser</u>
                                                   e = Button(parent=scene, model=m, collider='mesh', color=color.red,
<u>FileBrowserSave</u>
                                                   highlight_color=color.yellow)
<u>FirstPersonController</u>
FrameAnimation3d
                                                  EditorCamera()
<u>GridEditor</u>
<u>PixelEditor</u>
ASCIIEditor
```

<u>HealthBar</u> <u>HotReloader</u> #

```
light dark
Entity
<u>Text</u>
Button
mouse
<u>raycaster</u>
<u>application</u>
build
camera
color
curve
<u>duplicate</u>
input handler
<u>main</u>
mesh_importer
<u>scene</u>
shader
string_utilities
text
<u>texture_importer</u>
<u>ursinamath</u>
ursinastuff
<u>window</u>
<u>Audio</u>
Collider
BoxCollider
<u>SphereCollider</u>
<u>MeshCollider</u>
<u>CollisionZone</u>
Color
CubicBezier
<u>HitInfo</u>
<u>Keys</u>
<u>Light</u>
<u>DirectionalLight</u>
PointLight
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
<u>Mesh</u>
<u>Wait</u>
<u>Func</u>
<u>Sequence</u>
Shader
<u>Texture</u>
<u>Trigger</u>
<u>Empty</u>
LoopingList
Vec2
<u>Vec3</u>
Animation
Animator
<u>ButtonGroup</u>
ButtonList
Node
Conversation
Cursor
DebugMenu
<u>Draggable</u>
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
<u>EditorCamera</u>
ExitButton
FileButton
<u>FileBrowser</u>
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
FrameAnimation3d
<u>GridEditor</u>
<u>PixelEditor</u>
ASCIIEditor
<u>HealthBar</u>
<u>HotReloader</u>
```

```
SphereCollider(Collider)
ursina.collider
```

```
SphereCollider(entity, center=(0,0,0), radius=.5)
shape = CollisionSphere(center[0], center[1], center[2], radius)
node_path = entity.attachNewNode(CollisionNode('CollisionNode'))
visible = False
 e = Entity(model='sphere', x=2)
 e.collider = 'box'
                         # add BoxCollider based on entity's bounds.
 e.collider = 'sphere'
                         # add SphereCollider based on entity's
 bounds.
 e.collider = 'mesh'
                        # add MeshCollider based on entity's bounds.
 e.collider = BoxCollider(e, center=Vec3(0,0,0), size=Vec3(1,1,1))
 add BoxCollider at custom positions and size.
 e.collider = SphereCollider(e, center=Vec3(0,0,0), radius=.75)
 add SphereCollider at custom positions and size
 e.collider = MeshCollider(e, mesh=e.model, center=Vec3(0,0,0))
 add MeshCollider with custom shape and center.
 m = Prismatoid(base_shape=Circle(6), thicknesses=(1, .5))
 e = Button(parent=scene, model=m, collider='mesh', color=color.red,
 highlight_color=color.yellow)
 EditorCamera()
```

## MeshCollider(Collider)

ursina.collider

```
MeshCollider(entity, mesh=None, center=(0,0,0))
```

```
node_path = entity.attachNewNode(CollisionNode('CollisionNode'))
node = self.node_path.node()
collision_polygons = list()
visible = False
```

```
e = Entity(model='sphere', x=2)
e.collider = 'box' # add BoxCollider based on entity's bounds.
e.collider = 'sphere'
                      # add SphereCollider based on entity's
bounds.
e.collider = 'mesh' # add MeshCollider based on entity's bounds.
e.collider = BoxCollider(e, center=Vec3(0,0,0), size=Vec3(1,1,1))
add BoxCollider at custom positions and size.
e.collider = SphereCollider(e, center=Vec3(0,0,0), radius=.75)
add SphereCollider at custom positions and size.
e.collider = MeshCollider(e, mesh=e.model, center=Vec3(0,0,0))
add MeshCollider with custom shape and center.
m = Prismatoid(base_shape=Circle(6), thicknesses=(1, .5))
e = Button(parent=scene, model=m, collider='mesh', color=color.red,
highlight_color=color.yellow)
EditorCamera()
```

## CollisionZone(Entity)

ursina.collision\_zone

```
CollisionZone(radius=2, target_entities=None, **kwargs)
```

```
radius = radius
entities_with_mesh_colliders = target_entities # defaults to all
entities with a mesh collider

update()
update_colliders()
```

```
This will only enable mesh colliders' collision polygons within a
                                                       certain range,
<u>light</u> <u>dark</u>
                                                       in order to improve performance.
Entity
                                                        from ursina.shaders import basic_lighting_shader
<u>Text</u>
Button
                                                       window.vsync = False
mouse
                                                       application.asset_folder = application.asset_folder.parent
<u>raycaster</u>
                                                       terrain = Entity(model='heightmap_test', scale=32, texture='grass',
<u>application</u>
                                                       collider='mesh', shader=basic_lighting_shader)
                                                       from ursina.prefabs.first_person_controller import
build
camera
                                                       FirstPersonController
color
                                                        player = FirstPersonController(speed=10)
                                                       collision_zone = CollisionZone(parent=player, radius=32)
curve
<u>duplicate</u>
input handler
                                                        def input(key):
<u>main</u>
                                                            if key == 'c':
mesh_importer
<u>scene</u>
                                                                 terrain.collision = not terrain.collision
<u>shader</u>
string_utilities
<u>text</u>
<u>texture_importer</u>
                                                        Sky()
<u>ursinamath</u>
                                                       base.set_frame_rate_meter(True)
<u>ursinastuff</u>
<u>window</u>
<u>Audio</u>
Collider
                                                   Color(Vec4)
<u>BoxCollider</u>
                                                     ursina.color
<u>SphereCollider</u>
<u>MeshCollider</u>
                                                     Color(self,*p)
<u>CollisionZone</u>
Color
                                                     name
CubicBezier
                                                     r
<u>HitInfo</u>
                                                     g
<u>Keys</u>
                                                     b
<u>Light</u>
                                                     а
DirectionalLight
                                                     hsv
PointLight
                                                     h
<u>AmbientLight</u>
                                                     S
<u>SpotLight</u>
<u>Ursina</u>
                                                     brightness
<u>MeshModes</u>
<u>Mesh</u>
                                                     invert()
<u>Wait</u>
                                                     tint(amount)
<u>Func</u>
                                                     hsv(h, s, v, a=1)
<u>Sequence</u>
                                                     rgba(r, g, b, a=255)
Shader
                                                     rgb(r, g, b, a=255)
to_hsv(color)
<u>Texture</u>
<u>Trigger</u>
                                                     hex(value)
<u>Empty</u>
                                                     brightness(color)
LoopingList
                                                     inverse(color)
Vec2
                                                     random_color()
<u>Vec3</u>
                                                     tint(color, amount=.2)
Animation
                                                       print(color.brightness(color.blue))
Animator
                                                       print(_3)
ButtonGroup
ButtonList
                                                        p = Entity(x=-2)
<u>Node</u>
                                                        for key in color.colors:
Conversation
                                                             print(key)
Cursor
                                                             b = Button(parent=p, model=Quad(subdivisions=2),
DebugMenu
                                                        color=color.colors[key], text=key)
Draggable
                                                             b.text_entity.scale *= .5
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
                                                       grid layout(p.children, max x=8)
<u>EditorCamera</u>
<u>ExitButton</u>
                                                        for name in ('r', 'g', 'b', 'h', 's', 'v', 'brightness'):
    print(name + ':', getattr(color.random_color(), name))
FileButton
<u>FileBrowser</u>
<u>FileBrowserSave</u>
                                                        e = Entity(model='cube', color=color.lime)
<u>FirstPersonController</u>
                                                       print(e.color.name)
FrameAnimation3d
<u>GridEditor</u>
<u>PixelEditor</u>
```

ASCIIEditor HealthBar HotReloader

```
CubicBezier
                                                    ursina.curve
<u>light</u> <u>dark</u>
                                                    CubicBezier(a, b, c, d)
Entity
<u>Text</u>
                                                    b = b
Button
                                                    C = C
mouse
                                                    d = d
<u>raycaster</u>
                                                    cx = 3.0 * a
                                                    bx = 3.0 * (c - a) - self.cx
<u>application</u>
                                                    ax = 1.0 - self.cx - self.bx
<u>build</u>
                                                    cy = 3.0 * b
camera
                                                    by = 3.0 * (d - b) - self.cy
color
                                                    ay = 1.0 - self.cy - self.by
curve
<u>duplicate</u>
                                                    sample_curve_x(t)
input handler
                                                    sample_curve_y(t)
<u>main</u>
                                                    sample_curve_derivative_x(t)
mesh_importer
<u>scene</u>
                                                    calculate(x, epsilon=.0001)
                                                    solve_curve_x(t, epsilon=.0001)
<u>shader</u>
string_utilities
                                                     '''Draws a sheet with every curve and its name'''
<u>text</u>
<u>texture_importer</u>
                                                      camera.orthographic = True
<u>ursinamath</u>
                                                      camera.fov = 16
<u>ursinastuff</u>
                                                      camera.position = (9, 6)
<u>window</u>
                                                      window.color = color.black
<u>Audio</u>
                                                      i = 0
Collider
                                                      for e in dir(curve):
<u>BoxCollider</u>
                                                           try:
<u>SphereCollider</u>
                                                                item = getattr(curve, e)
print(item.__name__, ':'
<u>MeshCollider</u>
                                                                                             , item(.75))
<u>CollisionZone</u>
                                                                curve renderer = Entity(
Color
                                                                     model=Mesh(vertices=[Vec3(i / 31, item(i / 31), 0) for i
CubicBezier
                                                      in range(32)], mode='line', thickness=2),
<u>HitInfo</u>
                                                                    color=color.light_gray)
<u>Keys</u>
                                                                row = floor(i / 8)
<u>Light</u>
                                                                curve\_renderer.x = (i \% 8) * 2.5
DirectionalLight
                                                                curve_renderer.y = row * 1.75
<u>PointLight</u>
                                                                label = Text(parent=curve_renderer, text=item.__name__,
<u>AmbientLight</u>
                                                      scale=8, color=color.gray, y=-.1)
<u>SpotLight</u>
                                                                i += 1
<u>Ursina</u>
                                                           except:
<u>MeshModes</u>
                                                                pass
<u>Mesh</u>
<u>Wait</u>
                                                      c = CubicBezier(0, .5, 1, .5)
<u>Func</u>
                                                      print('----', c.calculate(.23))
<u>Sequence</u>
Shader
                                                      window.exit_button.visible = False
<u>Texture</u>
                                                      window.fps counter.enabled = False
<u>Trigger</u>
<u>Empty</u>
                                                      These are used by Entity when animating, like this:
LoopingList
Vec2
                                                      e = Entity()
<u>Vec3</u>
                                                      e.animate_y(1, curve=curve.in_expo)
Animation
                                                      e2 = Entity(x=1.5)
Animator
                                                      e2.animate_y(1, curve=curve.CubicBezier(0,.7,1,.3))
<u>ButtonGroup</u>
ButtonList
<u>Node</u>
Conversation
Cursor
DebugMenu
                                                  HitInfo
Draggable
<u>DropdownMenuButton</u>
                                                    ursina.hit info
<u>DropdownMenu</u>
<u>EditorCamera</u>
                                                    HitInfo(**kwargs)
<u>ExitButton</u>
FileButton
                                                    hit = None
<u>FileBrowser</u>
                                                    entity = None
<u>FileBrowserSave</u>
                                                    point = None
<u>FirstPersonController</u>
```

world\_point = None

normal = None

entities = []

hits = []

distance = math.inf

world\_normal = None

FrameAnimation3d

<u>GridEditor</u>

<u>PixelEditor</u>

**ASCIIEditor** 

<u>HotReloader</u>

<u>HealthBar</u>

Entity Text Button mouse

<u>raycaster</u>

application build camera color curve

duplicate input\_handler

main
mesh\_importer

scene
shader
string\_utilities
text

texture\_importer

ursinamath ursinastuff window

Audio
Collider
BoxCollider
SphereCollider
MeshCollider
CollisionZone
Color
CubicBezier

CubicBezier
HitInfo
Keys
Light

<u>DirectionalLight</u>

PointLight
AmbientLight
SpotLight
Ursina
MeshModes

Mesh Wait

<u>Func</u> <u>Sequence</u> <u>Shader</u>

<u>Texture</u> <u>Trigger</u>

Empty LoopingList

Vec2 Vec3

Animation Animator ButtonGroup ButtonList

Node Conversation

Cursor
DebugMenu
Draggable
DrondownMen

<u>DropdownMenuButton</u>

DropdownMenu
EditorCamera
ExitButton
FileButton
FileBrowser
FileBrowserSave

<u>FirstPersonController</u>

FrameAnimation3d GridEditor

PixelEditor ASCIIEditor HealthBar

<u>HotReloader</u>

# Keys(Enum)

ursina.input\_handler

left\_mouse\_down = 'left mouse down'
left\_mouse\_up = 'left mouse up' middle\_mouse\_down = 'middle mouse down' middle\_mouse\_up = 'middle mouse up' right\_mouse\_down = 'right mouse down' right\_mouse\_up = 'right mouse up' scroll\_up = 'scroll up' scroll\_down = 'scroll down' left\_arrow = 'left arrow' left\_arrow\_up = 'left arrow up' up\_arrow = 'up arrow'
up\_arrow\_up = 'up arrow up' down\_arrow = 'down arrow' down\_arrow\_up = 'down arrow up'
right\_arrow = 'right arrow'
right\_arrow\_up = 'right arrow up' left\_control = 'left control' right control = 'right control' left\_shift = 'left shift' right\_shift = 'right shift' left\_alt = 'left alt'
right\_alt = 'right alt'
left\_control\_up = 'left control up'
right\_control\_up = 'right control up'
left\_shift\_up = 'left shift up' right\_shift\_up = 'right shift up' left\_alt\_up = 'left alt up' right\_alt\_up = 'right alt up'
page\_down = 'page down' page\_down\_up = 'page down up' page\_up = 'page up' page\_up\_up = 'page up up' enter = 'enter' backspace = 'backspace' escape = 'escape' tab = 'tab'  $gamepad_a = 'gamepad a'$ gamepad\_a\_up = 'gamepad a up'
gamepad\_b = 'gamepad b'  $gamepad_b_up = 'gamepad b up'$  $gamepad_x = 'gamepad x'$ gamepad\_x\_up = 'gamepad x up'
gamepad\_y = 'gamepad y' gamepad\_y\_up = 'gamepad y up' gamepad left stick = 'gamepad left stick' gamepad\_left\_stick\_up = 'gamepad left stick up' gamepad\_right\_stick = 'gamepad right stick' gamepad\_right\_stick\_up = 'gamepad right stick up' gamepad\_back = 'gamepad back' gamepad\_back\_up = 'gamepad back up' gamepad\_start = 'gamepad start' gamepad\_dpad\_down = 'gamepad dpad down' gamepad\_dpad\_down\_up = 'gamepad dpad down up' gamepad\_dpad\_up = 'gamepad dpad up' gamepad\_dpad\_up\_up = 'gamepad dpad up up'
gamepad\_dpad\_left = 'gamepad dpad left' gamepad\_dpad\_left\_up = 'gamepad dpad left up' gamepad\_dpad\_right = 'gamepad dpad right' gamepad\_dpad\_right\_up = 'gamepad dpad right up' gamepad\_left\_shoulder = 'gamepad left shoulder' gamepad\_left\_shoulder\_up = 'gamepad left shoulder up' gamepad\_right\_shoulder = 'gamepad right shoulder' gamepad\_right\_shoulder\_up = 'gamepad right shoulder up'



<u>ursina.lights</u>

```
color
light dark
Entity
                                                   from ursina.shaders import lit_with_shadows_shader # you have to
                                                   apply this shader to enties for them to receive shadows.
<u>Text</u>
                                                   EditorCamera()
Button
                                                   Entity(model='plane', scale=10, color=color.gray,
mouse
<u>raycaster</u>
                                                   shader=lit_with_shadows_shader)
                                                   Entity(model='cube', y=1, shader=lit_with_shadows_shader)
<u>application</u>
                                                   pivot = Entity()
                                                   DirectionalLight(parent=pivot, y=2, z=3, shadows=True)
build
camera
color
curve
<u>duplicate</u>
input handler
                                               DirectionalLight(Light)
<u>main</u>
                                                 ursina.lights
mesh_importer
<u>scene</u>
                                                 DirectionalLight(shadows=True, **kwargs)
<u>shader</u>
string_utilities
                                                 shadow map resolution = Vec2(1024, 1024)
text
                                                 shadows
<u>texture_importer</u>
<u>ursinamath</u>
<u>ursinastuff</u>
                                                   from ursina.shaders import lit_with_shadows_shader # you have to
<u>window</u>
                                                   apply this shader to enties for them to receive shadows.
                                                   EditorCamera()
<u>Audio</u>
                                                   Entity(model='plane', scale=10, color=color.gray,
Collider
                                                   shader=lit_with_shadows_shader)
BoxCollider
                                                   Entity(model='cube', y=1, shader=lit_with_shadows_shader)
<u>SphereCollider</u>
                                                   pivot = Entity()
<u>MeshCollider</u>
                                                   DirectionalLight(parent=pivot, y=2, z=3, shadows=True)
<u>CollisionZone</u>
Color
CubicBezier
<u>HitInfo</u>
<u>Keys</u>
                                               PointLight(Light)
<u>Light</u>
<u>DirectionalLight</u>
                                                 ursina.lights
PointLight
<u>AmbientLight</u>
                                                 PointLight(**kwargs)
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
<u>Mesh</u>
                                                   from ursina.shaders import lit_with_shadows_shader # you have to
<u>Wait</u>
                                                   apply this shader to enties for them to receive shadows.
<u>Func</u>
                                                   EditorCamera()
<u>Sequence</u>
                                                   Entity(model='plane', scale=10, color=color.gray,
Shader
                                                   shader=lit_with_shadows_shader)
<u>Texture</u>
                                                   Entity(model='cube', y=1, shader=lit_with_shadows_shader)
<u>Trigger</u>
                                                   pivot = Entity()
<u>Empty</u>
                                                   DirectionalLight(parent=pivot, y=2, z=3, shadows=True)
LoopingList
Vec2
<u>Vec3</u>
Animation
                                               AmbientLight(Light)
Animator
                                                 ursina.lights
<u>ButtonGroup</u>
ButtonList
                                                 AmbientLight(**kwargs)
<u>Node</u>
Conversation
Cursor
DebugMenu
                                                   from ursina.shaders import lit_with_shadows_shader # you have to
<u>Draggable</u>
                                                   apply this shader to enties for them to receive shadows.
<u>DropdownMenuButton</u>
                                                   EditorCamera()
<u>DropdownMenu</u>
                                                   Entity(model='plane', scale=10, color=color.gray,
<u>EditorCamera</u>
                                                   shader=lit with shadows shader)
<u>ExitButton</u>
                                                   Entity(model='cube', y=1, shader=lit_with_shadows_shader)
FileButton
                                                   pivot = Entity()
<u>FileBrowser</u>
                                                   DirectionalLight(parent=pivot, y=2, z=3, shadows=True)
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
FrameAnimation3d
<u>GridEditor</u>
<u>PixelEditor</u>
                                               SpotLight(Light)
ASCIIEditor
```

<u>HealthBar</u> <u>HotReloader</u> Light(\*\*kwargs)

```
ursina.lights
                                                   SpotLight(**kwargs)
light dark
Entity
                                                     from ursina.shaders import lit_with_shadows_shader # you have to
<u>Text</u>
Button
                                                      apply this shader to enties for them to receive shadows.
mouse
                                                     EditorCamera()
                                                     Entity(model='plane', scale=10, color=color.gray,
<u>raycaster</u>
                                                     shader=lit_with_shadows_shader)
<u>application</u>
                                                     Entity(model='cube', y=1, shader=lit_with_shadows_shader)
                                                     pivot = Entity()
build
                                                     DirectionalLight(parent=pivot, y=2, z=3, shadows=True)
camera
color
curve
<u>duplicate</u>
input handler
<u>main</u>
                                                 Ursina()
mesh_importer
                                                   ursina.main
<u>scene</u>
shader
                                                   Ursina(**kwargs): # optional arguments: title, fullscreen, size,
string_utilities
                                                   forced_aspect_ratio, position, vsync, borderless, show_ursina_splash,
render_mode, development_mode, editor_ui_enabled
text
<u>texture_importer</u>
<u>ursinamath</u>
                                                   mouse = mouse
ursinastuff
<u>window</u>
                                                   input_up(key)
                                                   input_hold(key)
<u>Audio</u>
                                                   input(key)
Collider
                                                   run()
<u>BoxCollider</u>
<u>SphereCollider</u>
                                                     app = Ursina()
<u>MeshCollider</u>
                                                     app.run()
<u>CollisionZone</u>
Color
CubicBezier
HitInfo
<u>Keys</u>
                                                 MeshModes (Enum)
<u>Light</u>
DirectionalLight
                                                   ursina.mesh
PointLight
<u>AmbientLight</u>
                                                   triangle = 'triangle'
<u>SpotLight</u>
                                                   ngon = 'ngon'
<u>Ursina</u>
                                                   quad = 'quad'
line = 'line'
<u>MeshModes</u>
<u>Mesh</u>
                                                   point = 'point'
<u>Wait</u>
                                                   tristrip = 'tristrip'
<u>Func</u>
<u>Sequence</u>
<u>Shader</u>
<u>Texture</u>
<u>Trigger</u>
<u>Empty</u>
                                                 Mesh()
LoopingList
                                                   ursina.mesh
Vec2
Vec3
                                                   Mesh(vertices=None, triangles=None, colors=None, uvs=None,
Animation
                                                   normals=None, static=True, mode='triangle', thickness=1,
                                                   render_points_in_3d=True)
Animator
ButtonGroup
                                                   name = 'mesh'
ButtonList
                                                   vertices = vertices
Node
                                                   triangles = triangles
Conversation
                                                   colors = colors
Cursor
DebugMenu
                                                   uvs = uvs
                                                   normals = normals
Draggable
<u>DropdownMenuButton</u>
                                                   static = static
                                                   mode = mode
<u>DropdownMenu</u>
                                                   thickness = thickness
EditorCamera
                                                   render_points_in_3d = render_points_in_3d
ExitButton
FileButton
<u>FileBrowser</u>
                                                   generate()
                                                                  # call this after setting some of the variables to update
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
                                                   generate normals(smooth=True)
FrameAnimation3d
                                                   colorize(left=color.white, right=color.blue, down=color.red,
<u>GridEditor</u>
```

world\_space=True, strength=1)

project\_uvs(aspect\_ratio=1, direction='forward')

<u>PixelEditor</u> ASCIIEditor

<u>HealthBar</u> <u>HotReloader</u> up=color.green, back=color.white, forward=color.white, smooth=True,

```
save(name='', path=application.compressed_models_folder)
<u>light</u> <u>dark</u>
                                                      verts = ((0,0,0), (1,0,0), (.5, 1, 0), (-.5,1,0))
                                                      tris = (1, 2, 0, 2, 3, 0)
                                                      uvs = ((1.0, 0.0), (0.0, 1.0), (0.0, 0.0), (1.0, 1.0))
Entity
                                                      norms = ((0,0,-1),) * len(verts)
<u>Text</u>
Button
                                                      colors = (color.red, color.blue, color.lime, color.black)
mouse
<u>raycaster</u>
                                                      e = Entity(model=Mesh(vertices=verts, triangles=tris, uvs=uvs,
<u>application</u>
                                                      normals=norms, colors=colors), scale=2)
                                                      verts = (Vec3(0,0,0), Vec3(0,1,0), Vec3(1,1,0), Vec3(2,2,0),
<u>build</u>
camera
                                                      Vec3(0,3,0), Vec3(-2,3,0))
                                                      tris = ((0,1), (3,4,5))
color
curve
<u>duplicate</u>
                                                      lines = Entity(model=Mesh(vertices=verts, triangles=tris,
                                                      mode='line', thickness=4), color=color.cyan, z=-1)
input handler
                                                      points = Entity(model=Mesh(vertices=verts, mode='point',
<u>main</u>
                                                      thickness=.05), color=color.red, z=-1.01)
mesh_importer
<u>scene</u>
<u>shader</u>
string_utilities
<u>text</u>
<u>texture_importer</u>
                                                 Wait()
<u>ursinamath</u>
                                                    ursina.sequence
<u>ursinastuff</u>
<u>window</u>
                                                    Wait(duration)
<u>Audio</u>
                                                    duration = duration
Collider
<u>BoxCollider</u>
<u>SphereCollider</u>
                                                      e = Entity(model='quad')
<u>MeshCollider</u>
                                                      s = Sequence(
<u>CollisionZone</u>
                                                          1,
Color
                                                          Func(print, 'one'),
CubicBezier
                                                           Func(e.fade_out, duration=1),
<u>HitInfo</u>
<u>Keys</u>
                                                          Func(print, 'two'),
<u>Light</u>
                                                           Func(e.fade_in, duration=1),
DirectionalLight
                                                           loop=True
PointLight
                                                           )
<u>AmbientLight</u>
<u>SpotLight</u>
                                                      s.append(
<u>Ursina</u>
                                                           Func(print, 'appended to sequence')
<u>MeshModes</u>
<u>Mesh</u>
<u>Wait</u>
                                                      def input(key):
<u>Func</u>
                                                           actions = {'s' : s.start, 'f' : s.finish, 'p' : s.pause, 'r' :
<u>Sequence</u>
                                                      s.resume}
Shader
                                                           if key in actions:
<u>Texture</u>
                                                               actions[key]()
<u>Trigger</u>
<u>Empty</u>
LoopingList
Vec2
<u>Vec3</u>
                                                  Func()
Animation
                                                    ursina.sequence
Animator
ButtonGroup
                                                    Func(func, *args, **kwargs)
ButtonList
<u>Node</u>
                                                    func = func
Conversation
                                                    args = args
Cursor
                                                    kwargs = kwargs
DebugMenu
                                                    delay = 0
<u>Draggable</u>
                                                    finished = False
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
<u>EditorCamera</u>
                                                      e = Entity(model='quad')
<u>ExitButton</u>
                                                      s = Sequence(
FileButton
                                                          1,
<u>FileBrowser</u>
                                                           Func(print, 'one'),
<u>FileBrowserSave</u>
                                                           Func(e.fade_out, duration=1),
<u>FirstPersonController</u>
FrameAnimation3d
                                                           Func(print, 'two'),
<u>GridEditor</u>
                                                           Func(e.fade_in, duration=1),
<u>PixelEditor</u>
                                                           loop=True
ASCIIEditor
<u>HealthBar</u>
```

<u>HotReloader</u>

clear(regenerate=True)

```
s.append(
                                                           Func(print, 'appended to sequence')
light dark
                                                      def input(key):
                                                           actions = {'s' : s.start, 'f' : s.finish, 'p' : s.pause, 'r' :
Entity
<u>Text</u>
                                                      s.resume}
Button
                                                           if key in actions:
mouse
                                                                actions[key]()
<u>raycaster</u>
<u>application</u>
build
camera
                                                  Sequence()
color
                                                    ursina.sequence
curve
<u>duplicate</u>
                                                    Sequence(*args, **kwargs)
input handler
<u>main</u>
                                                    args = list(args)
mesh_importer
                                                    t = 0
<u>scene</u>
                                                    time_step = Sequence.default_time_step
shader
                                                    duration = 0
string_utilities
                                                    funcs = list()
<u>text</u>
                                                    paused = True
<u>texture_importer</u>
                                                    loop = False
<u>ursinamath</u>
                                                    auto destroy = True
ursinastuff
                                                    finished
<u>window</u>
                                                    generate()
<u>Audio</u>
                                                    append(arg)
Collider
                                                    extend(list)
<u>BoxCollider</u>
                                                    start()
<u>SphereCollider</u>
                                                    pause()
<u>MeshCollider</u>
                                                    resume()
<u>CollisionZone</u>
                                                    finish()
Color
                                                    kill()
CubicBezier
                                                    update()
<u>HitInfo</u>
<u>Keys</u>
                                                      e = Entity(model='quad')
<u>Light</u>
                                                       s = Sequence(
DirectionalLight
PointLight
                                                           Func(print, 'one'),
<u>AmbientLight</u>
                                                           Func(e.fade_out, duration=1),
<u>SpotLight</u>
<u>Ursina</u>
                                                           Func(print, 'two'),
<u>MeshModes</u>
                                                           Func(e.fade_in, duration=1),
<u>Mesh</u>
                                                           loop=True
<u>Wait</u>
<u>Func</u>
<u>Sequence</u>
                                                      s.append(
<u>Shader</u>
                                                           Func(print, 'appended to sequence')
<u>Texture</u>
<u>Trigger</u>
<u>Empty</u>
                                                      def input(key):
<u>LoopingList</u>
                                                           actions = {'s' : s.start, 'f' : s.finish, 'p' : s.pause, 'r' :
Vec2
                                                      s.resume}
<u>Vec3</u>
                                                           if key in actions:
                                                                actions[key]()
Animation
Animator
<u>ButtonGroup</u>
ButtonList
<u>Node</u>
                                                  Shader
Conversation
Cursor
                                                    ursina.shader
DebugMenu
Draggable
                                                    Shader(language=Panda3dShader.SL_GLSL, vertex=default_vertex_shader,
<u>DropdownMenuButton</u>
                                                    fragment=default_fragment_shader, geometry='', **kwargs)
<u>DropdownMenu</u>
EditorCamera
                                                    path = Path(_caller.filename)
ExitButton
                                                    language = language
FileButton
                                                    vertex = vertex
<u>FileBrowser</u>
                                                    fragment = fragment
<u>FileBrowserSave</u>
                                                    geometry = geometry
<u>FirstPersonController</u>
                                                    entity = None
FrameAnimation3d
                                                    default_input = dict()
<u>GridEditor</u>
                                                    compiled = False
<u>PixelEditor</u>
ASCIIEditor
                                                    compile()
<u>HealthBar</u>
<u>HotReloader</u>
```

```
from time import perf_counter
                                                     t = perf_counter()
                                                     Entity(model='cube', shader=Shader())
light dark
                                                     EditorCamera()
                                                     print('tttttttttttt', perf_counter() - t)
Entity
                                                     def input(key):
                                                          if held_keys['control'] and key == 'r':
<u>Text</u>
Button
                                                              reload_shaders()
mouse
                                                     def reload_shaders():
<u>raycaster</u>
                                                          for e in scene.entities:
                                                              if hasattr(e, '_shader'):
    print('-----', e.shader)
<u>application</u>
build
camera
color
curve
<u>duplicate</u>
input handler
                                                 Texture()
<u>main</u>
                                                   ursina.texture
mesh_importer
<u>scene</u>
                                                   Texture(value)
<u>shader</u>
string_utilities
                                                   filtering = Texture.default_filtering
default: 'bilinear'
                                                                                                # None/'bilinear'/'mipmap'
text
<u>texture_importer</u>
                                                   name
<u>ursinamath</u>
                                                   size
ursinastuff
                                                   width
<u>window</u>
                                                   height
                                                   pixels
<u>Audio</u>
                                                   repeat
Collider
<u>BoxCollider</u>
                                                   get_pixel(x, y)
<u>SphereCollider</u>
                                                   get_pixels(start, end)
<u>MeshCollider</u>
                                                   set_pixel(x, y, color)
<u>CollisionZone</u>
                                                   apply()
Color
                                                   save(path)
CubicBezier
HitInfo
<u>Keys</u>
                                                          The Texture class rarely used manually but usually instantiated
<u>Light</u>
                                                          when assigning a texture to an Entity
DirectionalLight
                                                         texture = Texture(path / PIL.Image / panda3d.core.Texture)
PointLight
<u>AmbientLight</u>
                                                          A texture file can be a .png, .jpg or .psd.
<u>SpotLight</u>
                                                         If it's a .psd it and no compressed version exists, it will
<u>Ursina</u>
                                                     compress it automatically.
<u>MeshModes</u>
<u>Mesh</u>
                                                     e = Entity(model='quad', texture='brick')
<u>Wait</u>
                                                     e.texture.set_pixel(0, 2, color.blue)
<u>Func</u>
                                                    e.texture.apply()
<u>Sequence</u>
<u>Shader</u>
<u>Texture</u>
<u>Trigger</u>
<u>Empty</u>
                                                 Trigger(Entity)
LoopingList
Vec2
                                                   ursina.trigger
<u>Vec3</u>
                                                   Trigger(**kwargs)
Animation
Animator
                                                   trigger_targets = None
<u>ButtonGroup</u>
                                                   radius = .5
ButtonList
                                                   triggerers = list()
<u>Node</u>
                                                   update rate = 4
Conversation
Cursor
                                                   update()
DebugMenu
<u>Draggable</u>
                                                     player = Entity(model='cube', color=color.azure, scale=.05)
<u>DropdownMenuButton</u>
                                                     def update():
<u>DropdownMenu</u>
                                                          player.x += held_keys['d'] * time.dt * 2
EditorCamera
                                                          player.x -= held_keys['a'] * time.dt * 2
ExitButton
FileButton
                                                     t = Trigger(trigger_targets=(player,), x=1, model='sphere',
<u>FileBrowser</u>
                                                     color=color(0,1,1,.5))
<u>FileBrowserSave</u>
                                                     t.on_trigger_enter = Func(print, 'enter')
<u>FirstPersonController</u>
                                                     t.on_trigger_exit = Func(print, 'exit')
FrameAnimation3d
                                                    t.on_trigger_stay = Func(print, 'stay')
<u>GridEditor</u>
<u>PixelEditor</u>
ASCIIEditor
```

<u>HealthBar</u> <u>HotReloader</u>

```
Empty()
                                                         ursinastuff
<u>light</u> <u>dark</u>
                                                         Empty(**kwargs)
Entity
<u>Text</u>
                                                         invoke(function, *args, **kwargs)
Button
                                                         destroy(entity, delay=0)
mouse
                                                         find_sequence(name, file_types, folders) # find frame_0, frame_1,
<u>raycaster</u>
                                                         frame_2 and so on
                                                         import_all_classes(path=application.asset_folder, debug=False)
<u>application</u>
                                                         print_on_screen(text, position=window.top_left, origin=(-.5,.5),
<u>build</u>
                                                         scale=1, duration=1)
camera
color
                                                           def test func(item, x=None, y=None):
curve
                                                                print(item, x, y)
<u>duplicate</u>
input handler
                                                           test_func('test')
<u>main</u>
                                                           invoke(test_func, 'test', delay=.1)
invoke(test_func, 'test1', 1, 2, delay=.2)
invoke(test_func, 'test2', x=1, y=2, delay=.3)
mesh_importer
<u>scene</u>
<u>shader</u>
string_utilities
                                                           def input(key):
<u>text</u>
                                                                if key == 'space':
<u>texture_importer</u>
                                                                      print_on_screen('debug message', position=(0,0), origin=
<u>ursinamath</u>
                                                           (0,0), scale=2)
<u>ursinastuff</u>
<u>window</u>
<u>Audio</u>
Collider
<u>BoxCollider</u>
                                                       LoopingList(list)
<u>SphereCollider</u>
                                                         ursinastuff
<u>MeshCollider</u>
<u>CollisionZone</u>
                                                         LoopingList(**kwargs)
<u>Color</u>
CubicBezier
<u>HitInfo</u>
<u>Keys</u>
                                                           def test_func(item, x=None, y=None):
<u>Light</u>
                                                                print(item, x, y)
DirectionalLight
PointLight
                                                           test_func('test')
<u>AmbientLight</u>
                                                           invoke(test_func, 'test', delay=.1)
invoke(test_func, 'test1', 1, 2, delay=.2)
invoke(test_func, 'test2', x=1, y=2, delay=.3)
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
<u>Mesh</u>
                                                           def input(key):
<u>Wait</u>
                                                                 if key == 'space':
<u>Func</u>
                                                                      print_on_screen('debug message', position=(0,0), origin=
<u>Sequence</u>
                                                           (0,0), scale=2)
Shader
<u>Texture</u>
<u>Trigger</u>
<u>Empty</u>
LoopingList
                                                      Vec2(PandaVec2)
Vec2
<u>Vec3</u>
                                                         ursina.vec2
Animation
                                                         Vec2(**kwargs)
Animator
<u>ButtonGroup</u>
                                                         х
ButtonList
                                                         У
<u>Node</u>
Conversation
Cursor
                                                           a = Vec2(1,1)
DebugMenu
                                                           print(a)
<u>Draggable</u>
                                                         print(round(a))
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
<u>EditorCamera</u>
<u>ExitButton</u>
FileButton
                                                      Vec3(PandaVec3)
<u>FileBrowser</u>
                                                         ursina.vec3
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
                                                         Vec3(**kwargs)
FrameAnimation3d
<u>GridEditor</u>
                                                         Х
<u>PixelEditor</u>
                                                         У
ASCIIEditor
                                                         z
<u>HealthBar</u>
```

ху

<u>HotReloader</u>

```
ΧZ
                                                   уz
<u>light</u> <u>dark</u>
                                                     a = Vec3(1,0,0) * 2
                                                     a = Vec3(1,0,1) * Vec3(2,1,2)
Entity
                                                     b = Vec3(1.252352324,0,1)
<u>Text</u>
Button
                                                     b \leftarrow Vec3(0,1)
mouse
<u>raycaster</u>
<u>application</u>
                                                 Animation(Sprite)
build
camera
                                                   ursina.prefabs.animation
color
curve
                                                   Animation(name, fps=12, loop=True, autoplay=True, frame_times=None,
<u>duplicate</u>
                                                   **kwargs)
input handler
<u>main</u>
                                                   sequence = Sequence(loop=loop, auto_destroy=False)
mesh_importer
                                                   frame_times = frame_times
<u>scene</u>
                                                   is_playing = False
<u>shader</u>
                                                   autoplay = autoplay
string_utilities
                                                                          # get the duration of the animation. you can't set
                                                   duration
text
                                                   it. to do so, change the fps instead.
<u>texture_importer</u>
<u>ursinamath</u>
                                                   start()
ursinastuff
                                                   pause()
<u>window</u>
                                                   resume()
                                                   finish()
<u>Audio</u>
Collider
                                                    1.1.1
<u>BoxCollider</u>
                                                     Loads an image sequence as a frame animation.
<u>SphereCollider</u>
                                                     So if you have some frames named image_000.png, image_001.png,
<u>MeshCollider</u>
                                                     image_002.png and so on,
<u>CollisionZone</u>
                                                     you can load it like this: Animation('image')
Color
CubicBezier
                                                     You can also load a .gif by including the file type:
<u>HitInfo</u>
                                                     Animation('image.gif')
<u>Keys</u>
<u>Light</u>
DirectionalLight
                                                     Animation('ursina_wink')
PointLight
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
                                                 Animator()
<u>Mesh</u>
<u>Wait</u>
                                                   ursina.prefabs.animator
<u>Func</u>
<u>Sequence</u>
                                                   Animator(animations=None, start_state='')
<u>Shader</u>
<u>Texture</u>
                                                   animations = animations
                                                                                   # dict
<u>Trigger</u>
                                                   start_state = start_state
<u>Empty</u>
                                                   state = start_state
LoopingList
Vec2
<u>Vec3</u>
                                                     anim = Animation('ursina_wink', loop=True, autoplay=False)
                                                     a = Animator(
Animation
                                                          animations = {
Animator
                                                               'lol' : Entity(model='cube', color=color.red),
<u>ButtonGroup</u>
                                                               'yo' : Entity(model='cube', color=color.green, x=1),
ButtonList
                                                               'help' : anim,
<u>Node</u>
                                                          }
Conversation
Cursor
                                                     a.state = 'yo'
DebugMenu
<u>Draggable</u>
                                                     Text('press <red>1<default>, <green>2<default> or <violet>3<default>
<u>DropdownMenuButton</u>
                                                     to toggle different animator states', origin=(0, -.5), y=-.4)
<u>DropdownMenu</u>
<u>EditorCamera</u>
                                                     def input(key):
<u>ExitButton</u>
                                                          if key == '1':
FileButton
                                                              a.state = 'lol'
<u>FileBrowser</u>
                                                          if key == '2':
<u>FileBrowserSave</u>
                                                              a.state = 'yo'
<u>FirstPersonController</u>
```

if key == '3':

a.state = 'help'

print(anim.enabled)

FrameAnimation3d

<u>GridEditor</u>

<u>PixelEditor</u> **ASCIIEditor** <u>HealthBar</u> <u>HotReloader</u>

```
ButtonGroup(Entity)
                                                ursina.prefabs.button_group
light dark
                                                ButtonGroup(options=None, default='', min_selection=1,
Entity
                                                max_selection=1, **kwargs)
<u>Text</u>
Button
                                                deselected color = Button.color
                                                selected color = color.azure
<u>raycaster</u>
                                                min_selection = min_selection
                                                max_selection = max(min_selection, max_selection)
<u>application</u>
                                                buttons = list()
build
                                                selected = list()
camera
                                                options = options
color
                                                parent = camera.ui
curve
                                                scale = Text.size * 2
<u>duplicate</u>
                                                value
input handler
<u>main</u>
                                                layout()
mesh_importer
                                                input(key)
scene
                                                select(b)
shader
                                                on_value_changed()
string_utilities
text
                                                  gender_selection = ButtonGroup(('man', 'woman', 'other'))
<u>texture_importer</u>
                                                  on_off_switch = ButtonGroup(('off', 'on'), min_selection=1, y=-.1,
ursinamath
                                                  default='on', selected_color=color.red)
ursinastuff
<u>window</u>
                                                  def on value changed():
                                                      print('set gender:', gender_selection.value)
Audio
                                                  gender_selection.on_value_changed = on_value_changed
Collider
BoxCollider
                                                  def on value changed():
<u>SphereCollider</u>
                                                      print('turn:', on_off_switch.value)
<u>MeshCollider</u>
                                                  on_off_switch.on_value_changed = on_value_changed
<u>CollisionZone</u>
Color
                                                  window.color = color. 32
CubicBezier
HitInfo
<u>Keys</u>
<u>Light</u>
<u>DirectionalLight</u>
PointLight
                                              ButtonList(Entity)
<u>AmbientLight</u>
                                                ursina.prefabs.button_list
<u>SpotLight</u>
<u>Ursina</u>
                                                ButtonList(button_dict, button_height=1.1, fit_height=True, width=.5,
MeshModes
                                                **kwargs)
<u>Mesh</u>
Wait
                                                fit height = fit height
<u>Func</u>
                                                button height = button height
<u>Sequence</u>
                                                text_entity = Text(parent=self, origin=(-.5,.5), text='empty',
Shader
                                                world_scale=20, z=-.1, x=.01, line_height=button_height)
Texture
                                                button_height = self.text_entity.height
<u>Trigger</u>
                                                button_dict = button_dict
<u>Empty</u>
                                                highlight = Entity(parent=self, model='quad', color=color.white33,
LoopingList
                                                scale=(1,self.button_height), origin=(-.5,.5), z=-.01,
Vec2
                                                add_to_scene_entities=False)
Vec3
                                                selection_marker = Entity(parent=self, model='quad',
                                                color=color.azure, scale=(1,self.button_height), origin=(-.5,.5),
Animation
                                                z=-.02, enabled=False, add_to_scene_entities=False)
Animator
<u>ButtonGroup</u>
                                                input(key)
ButtonList
                                                update()
Node
                                                on disable()
<u>Conversation</u>
Cursor
                                                  default = Func(print, 'not yet implemented')
DebugMenu
<u>Draggable</u>
                                                  def test(a=1, b=2):
<u>DropdownMenuButton</u>
                                                      print('--
                                                                 ·---:', a, b)
DropdownMenu
EditorCamera
                                                  button_dict = {
<u>ExitButton</u>
                                                       'one':
                                                                   None,
FileButton
                                                      'two':
                                                                   default,
<u>FileBrowser</u>
                                                      'tree':
                                                                   Func(test, 3, 4),
<u>FileBrowserSave</u>
                                                       'four':
                                                                   Func(test, b=3, a=4),
<u>FirstPersonController</u>
FrameAnimation3d
                                                  for i in range(6, 20):
<u>GridEditor</u>
                                                      button_dict[f'button {i}'] = Func(print, i)
<u>PixelEditor</u>
ASCIIEditor
                                                  sound effects = {}
HealthBar
                                                  current_sound = None
```

<u>HotReloader</u>

```
light dark
Entity
<u>Text</u>
Button
mouse
<u>raycaster</u>
<u>application</u>
build
camera
color
curve
<u>duplicate</u>
input handler
<u>main</u>
mesh_importer
<u>scene</u>
shader
string_utilities
text
<u>texture_importer</u>
<u>ursinamath</u>
ursinastuff
<u>window</u>
<u>Audio</u>
Collider
BoxCollider
<u>SphereCollider</u>
<u>MeshCollider</u>
<u>CollisionZone</u>
Color
CubicBezier
HitInfo
<u>Keys</u>
<u>Light</u>
DirectionalLight
PointLight
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
<u>Mesh</u>
<u>Wait</u>
<u>Func</u>
<u>Sequence</u>
<u>Shader</u>
<u>Texture</u>
<u>Trigger</u>
<u>Empty</u>
LoopingList
Vec2
<u>Vec3</u>
Animation
Animator
ButtonGroup
ButtonList
Node
<u>Conversation</u>
Cursor
DebugMenu
<u>Draggable</u>
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
<u>EditorCamera</u>
ExitButton
FileButton
<u>FileBrowser</u>
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
FrameAnimation3d
<u>GridEditor</u>
<u>PixelEditor</u>
ASCIIEditor
HealthBar
<u>HotReloader</u>
```

```
for name in ('blip', 'boom', 'coin', 'hurt', 'jump', 'lose',
   'powerup', 'teleport'):
    for i in range(1,4):
        a = Audio(f'{name}_{i}', autoplay=False)
        if a.clip:
            sound_effects[f'{name}_{i}'] = Func(print, 'a')

bl = ButtonList(sound_effects)
```

#### Node

ursina.prefabs.conversation

```
Node(button_dict, button_height=1.1, fit_height=True, width=.5,
**kwargs)
```

```
app = Ursina()
    conversation = Conversation()
    bar_mission_solved = False
    convo = ''
I'm looking for my sister. Can you help me find her, please? I
haven't seen her in days! Who know what could've happened!?
I'm worried. Will you help me?
     'Yes, of course. This can be a dangerous city.
        Oh no! Do you think something happened to her?
        What should I do?!
            * She's probably fine. She can handle herself.
                You're right. I'm still worried though.
                    * Don't worry, I'll look for her.
            * Maybe. (chaos += 1)
                Help me look for her, please! *runs off*
    * I'm sorry, but I don't have time right now.
        A true friend wouldn't say that. (evil += 1)
    * I know where she is! (if bar_mission_solved)
        Really? Where?
            * I saw her on a ship by the docks, it looked like they
were ready to set off.
                Thank you! *runs off*
    conversation.start_conversation(convo)
    window.size = window.fullscreen size * .5
    Sprite('shore', z=1)
    app.run()
```

## Conversation(<a href="Entity">Entity</a>)

ursina.prefabs.conversation

#### Conversation(\*\*kwargs)

```
question = Button(
        parent=self,
        text origin=(-.5,0),
        scale x=1,
        scale y=.1,
        model=Quad(radius=.5, aspect=1/.1),
        text='What do you want\nWhat do you want?'
more_indicator = Entity(parent=self.question, model=Circle(3),
position=(.45,-.4,-.1), rotation_z=180, color=color.azure,
world_scale=.5, z=-1, enabled=False)
spacing = 4 * .02
wordwrap = 65
button_model = Quad(radius=.5, aspect=1/.075)
answer_0 = Button(parent=self, text='answer_0', y=self.question.y-
self.spacing-.025, scale=(1,.075), text_origin=(-.5,0),
model=copy(self.button model))
answer_1 = Button(parent=self, text='answer_1', y=self.answer_0.y-
self.spacing, scale=(1,.075), text_origin=(-.5,0),
model=copy(self.button_model))
```

answer 2 = Button(parent=self, text='answer 2', y=self.answer 1.y-

```
model=copy(self.button_model))
                                                  buttons = (self.answer_0, self.answer_1, self.answer_2)
                                                  question_appear_sequence = None
<u>light</u> <u>dark</u>
                                                  button_appear_sequence = None
Entity
                                                  started = False
<u>Text</u>
Button
                                                  toggle()
                                                  ask(node, question_part=0)
mouse
                                                  on_click(node=child)
<u>raycaster</u>
                                                  input(key)
<u>application</u>
                                                  next()
                                                  start_conversation(conversation)
build
camera
                                                  parse conversation(convo)
color
                                                    app = Ursina()
curve
<u>duplicate</u>
                                                         conversation = Conversation()
input handler
                                                         bar mission solved = False
                                                         convo = '
<u>main</u>
                                                    I'm looking for my sister. Can you help me find her, please? I
mesh_importer
<u>scene</u>
                                                    haven't seen her in days! Who know what could've happened!?
shader
                                                    I'm worried. Will you help me?
                                                           Yes, of course. This can be a dangerous city.
string_utilities
                                                              Oh no! Do you think something happened to her?
text
<u>texture_importer</u>
                                                              What should I do?!
                                                                   * She's probably fine. She can handle herself.
<u>ursinamath</u>
ursinastuff
                                                                       You're right. I'm still worried though.
                                                                            * Don't worry, I'll look for her.
<u>window</u>
                                                                   * Maybe. (chaos += 1)
                                                                       Help me look for her, please! *runs off*
<u>Audio</u>
                                                         * I'm sorry, but I don't have time right now.
Collider
                                                             A true friend wouldn't say that. (evil += 1)
<u>BoxCollider</u>
                                                         * I know where she is! (if bar_mission_solved)
<u>SphereCollider</u>
                                                              Really? Where?
<u>MeshCollider</u>
                                                                   * I saw her on a ship by the docks, it looked like they
<u>CollisionZone</u>
Color
                                                    were ready to set off.
CubicBezier
                                                                       Thank you! *runs off*
<u>HitInfo</u>
                                                         conversation.start conversation(convo)
<u>Keys</u>
<u>Light</u>
DirectionalLight
                                                         window.size = window.fullscreen_size * .5
                                                         Sprite('shore', z=1)
PointLight
<u>AmbientLight</u>
                                                         app.run()
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
<u>Mesh</u>
<u>Wait</u>
                                                Cursor(<u>Entity</u>)
<u>Func</u>
                                                  ursina.prefabs.cursor
<u>Sequence</u>
<u>Shader</u>
                                                  Cursor(**kwargs)
<u>Texture</u>
<u>Trigger</u>
                                                  parent = camera.ui
<u>Empty</u>
                                                  texture = 'cursor'
LoopingList
                                                  model = 'quad'
Vec2
                                                  color = color.light_gray
<u>Vec3</u>
                                                  render_queue = 1
Animation
                                                  update()
Animator
ButtonGroup
                                                    Button('button').fit_to_text()
ButtonList
                                                    Panel()
<u>Node</u>
                                                    camera.orthographic = True
<u>Conversation</u>
                                                    camera.fov = 100
Cursor
                                                    e = Entity(model='cube')
DebugMenu
                                                    mouse._mouse_watcher.setGeometry(e.model.node())
Draggable
                                                    mouse.visible = False
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
<u>EditorCamera</u>
<u>ExitButton</u>
FileButton
                                                DebugMenu(<u>Draggable</u>)
<u>FileBrowser</u>
<u>FileBrowserSave</u>
                                                  ursina.prefabs.debug_menu
<u>FirstPersonController</u>
FrameAnimation3d
                                                  DebugMenu(target, **kwargs)
<u>GridEditor</u>
<u>PixelEditor</u>
```

target = target

scale = (.2, .025)

text = '<orange>' + target.\_\_class\_\_.\_\_name\_

**ASCIIEditor** 

<u>HotReloader</u>

<u>HealthBar</u>

self.spacing, scale=(1,.075), text\_origin=(-.5,0),

```
draw_functions()
light dark
                                                  DebugMenu(Audio('night sky'))
Entity
<u>Text</u>
Button
mouse
                                                Draggable(Button)
<u>raycaster</u>
                                                  ursina.prefabs.draggable
<u>application</u>
                                                  Draggable(**kwargs)
build
camera
                                                  require_key = None
color
                                                  dragging = False
curve
                                                  delta_drag = 0
<u>duplicate</u>
                                                  start_pos = self.world_position
input handler
                                                  start_offset = (0,0,0)
<u>main</u>
                                                  step = (0,0,0)
mesh_importer
                                                  plane_direction = (0,0,1)
<u>scene</u>
                                                  lock = Vec3(0,0,0)
                                                                          # set to 1 to lock movement on any of x, y and
<u>shader</u>
                                                  z axes
string_utilities
                                                  min_x, self.min_y, self.min_z = -inf, -inf, -inf
text
                                                  max_x, self.max_y, self.max_z = inf, inf, inf
<u>texture_importer</u>
<u>ursinamath</u>
                                                  input(key)
<u>ursinastuff</u>
                                                  start_dragging()
<u>window</u>
                                                  stop_dragging()
                                                  update()
<u>Audio</u>
Collider
                                                    Entity(model='plane', scale=8, texture='white_cube', texture_scale=
<u>BoxCollider</u>
                                                    (8,8))
<u>SphereCollider</u>
                                                    draggable_button = Draggable(scale=.1, text='drag me', position=(-.5,
<u>MeshCollider</u>
<u>CollisionZone</u>
                                                    world_space_draggable = Draggable(parent=scene, model='cube',
Color
                                                    color=color.azure, plane_direction=(0,1,0), lock=(1,0,0))
CubicBezier
<u>HitInfo</u>
                                                    EditorCamera(rotation=(30,10,0))
<u>Keys</u>
                                                    world_space_draggable.drop = Func(print, 'dropped cube')
<u>Light</u>
<u>DirectionalLight</u>
PointLight
<u>AmbientLight</u>
<u>SpotLight</u>
                                                DropdownMenuButton(<u>Button</u>)
<u>Ursina</u>
<u>MeshModes</u>
                                                  ursina.prefabs.dropdown menu
<u>Mesh</u>
<u>Wait</u>
                                                  DropdownMenuButton(text='', **kwargs)
<u>Func</u>
<u>Sequence</u>
Shader
<u>Texture</u>
                                                    from ursina.prefabs.dropdown_menu import DropdownMenu,
<u>Trigger</u>
                                                    DropdownMenuButton
<u>Empty</u>
LoopingList
                                                    DropdownMenu('File', buttons=(
Vec2
                                                         DropdownMenuButton('New'),
<u>Vec3</u>
                                                         DropdownMenuButton('Open'),
                                                         DropdownMenu('Reopen Project', buttons=(
Animation
                                                             DropdownMenuButton('Project 1'),
Animator
                                                             DropdownMenuButton('Project 2'),
<u>ButtonGroup</u>
ButtonList
                                                         DropdownMenuButton('Save'),
<u>Node</u>
                                                         DropdownMenu('Options', buttons=(
<u>Conversation</u>
                                                             DropdownMenuButton('Option a'),
Cursor
                                                             DropdownMenuButton('Option b'),
DebugMenu
                                                             )),
<u>Draggable</u>
                                                         DropdownMenuButton('Exit'),
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
<u>EditorCamera</u>
<u>ExitButton</u>
FileButton
<u>FileBrowser</u>
                                                DropdownMenu(DropdownMenuButton)
<u>FileBrowserSave</u>
                                                  ursina.prefabs.dropdown menu
<u>FirstPersonController</u>
FrameAnimation3d
                                                  DropdownMenu(text='', buttons=list(), **kwargs)
<u>GridEditor</u>
<u>PixelEditor</u>
ASCIIEditor
                                                  position = window.top left
                                                  buttons = buttons
<u>HealthBar</u>
```

<u>HotReloader</u>

```
arrow_symbol = Text(world_parent=self, text='>', origin=(.5,.5),
                                                    position=(.95, 0), color=color.gray)
<u>light</u> <u>dark</u>
                                                    open()
                                                    close()
Entity
                                                    on_mouse_enter()
                                                    input(key)
<u>Text</u>
Button
                                                    update()
mouse
                                                      from ursina.prefabs.dropdown_menu import DropdownMenu,
<u>raycaster</u>
                                                      DropdownMenuButton
<u>application</u>
                                                      DropdownMenu('File', buttons=(
build
camera
                                                           DropdownMenuButton('New'),
                                                           DropdownMenuButton('Open'),
color
                                                           DropdownMenu('Reopen Project', buttons=(
curve
                                                                DropdownMenuButton('Project 1'),
<u>duplicate</u>
                                                                DropdownMenuButton('Project 2'),
input handler
<u>main</u>
                                                           DropdownMenuButton('Save'),
mesh_importer
                                                           DropdownMenu('Options', buttons=(
    DropdownMenuButton('Option a')
<u>scene</u>
shader
                                                                DropdownMenuButton('Option b'),
string_utilities
text
<u>texture_importer</u>
                                                           DropdownMenuButton('Exit'),
<u>ursinamath</u>
<u>ursinastuff</u>
<u>window</u>
<u>Audio</u>
Collider
                                                  EditorCamera(<u>Entity</u>)
<u>BoxCollider</u>
                                                    ursina.prefabs.editor camera
<u>SphereCollider</u>
<u>MeshCollider</u>
                                                    EditorCamera(**kwargs)
<u>CollisionZone</u>
Color
                                                    gizmo = Entity(parent=self, model='sphere', color=color.orange,
CubicBezier
                                                    scale=.025, add_to_scene_entities=False, enabled=False)
<u>HitInfo</u>
                                                    rotation_speed = 200
<u>Keys</u>
                                                    pan\_speed = Vec2(5, 5)
<u>Light</u>
                                                    move_speed = 10
DirectionalLight
                                                    zoom\_speed = 1.25
PointLight
                                                    zoom_smoothing = 8
<u>AmbientLight</u>
                                                    rotate_around_mouse_hit = False
<u>SpotLight</u>
                                                    start_position = self.position
<u>Ursina</u>
                                                    perspective_fov = camera.fov
<u>MeshModes</u>
                                                    orthographic_fov = camera.fov
<u>Mesh</u>
                                                    on destroy = self.on disable
<u>Wait</u>
                                                    hotkeys = {'toggle_orthographic':'shift+p', 'focus':'f',
<u>Func</u>
                                                    'reset_center':'shift+f'}
<u>Sequence</u>
Shader
                                                    on_enable()
<u>Texture</u>
                                                    on disable()
<u>Trigger</u>
                                                    input(key)
<u>Empty</u>
                                                    update()
LoopingList
Vec2
                                                      app = Ursina(vsync=False)
<u>Vec3</u>
                                                      Simple camera for debugging.
Animation
                                                      Hold right click and move the mouse to rotate around point.
Animator
<u>ButtonGroup</u>
ButtonList
                                                      sky = Sky()
Node
                                                      e = Entity(model='cube', color=color.white, collider='box')
<u>Conversation</u>
                                                      e.model.colorize()
Cursor
DebugMenu
                                                      from ursina.prefabs.first_person_controller import
<u>Draggable</u>
                                                      FirstPersonController
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
                                                      ground = Entity(model='plane', scale=32, texture='white_cube',
<u>EditorCamera</u>
                                                      texture_scale=(32,32), collider='box')
<u>ExitButton</u>
                                                      box = Entity(model='cube', collider='box', texture='white_cube',
scale=(10,2,2), position=(2,1,5), color=color.light_gray)
FileButton
<u>FileBrowser</u>
                                                      ec = EditorCamera(rotation_smoothing=2, enabled=1, rotation=
<u>FileBrowserSave</u>
                                                      (30,30,0))
<u>FirstPersonController</u>
FrameAnimation3d
                                                      rotation_info = Text(position=window.top_left)
<u>GridEditor</u>
<u>PixelEditor</u>
                                                      def update():
ASCIIEditor
                                                           rotation_info.text = str(int(ec.rotation_y)) + '\n' +
<u>HealthBar</u>
                                                      str(int(ec.rotation_x))
<u>HotReloader</u>
```

```
def input(key):
light dark
                                                        if key == 'tab': # press tab to toggle edit/play mode
                                                            ec.enabled = not ec.enabled
Entity
                                                            player.enabled = not player.enabled
<u>Text</u>
Button
mouse
<u>raycaster</u>
                                               ExitButton(Button)
<u>application</u>
                                                 ursina.prefabs.exit_button
build
camera
                                                 ExitButton(**kwargs)
color
curve
<u>duplicate</u>
                                                 on_click()
input handler
                                                 input(key)
<u>main</u>
mesh_importer
<u>scene</u>
                                                   This is the button in the upper right corner.
shader
                                                   You can click on it or press Shift+Q to close the program.
string_utilities
                                                   To disable it, set window.exit button.enabled to False
text
<u>texture_importer</u>
<u>ursinamath</u>
ursinastuff
<u>window</u>
                                               FileButton(<u>Button</u>)
<u>Audio</u>
Collider
                                                 ursina.prefabs.file browser
BoxCollider
<u>SphereCollider</u>
                                                 FileButton(load_menu, **kwargs)
<u>MeshCollider</u>
<u>CollisionZone</u>
                                                 load menu = load menu
Color
                                                 selected
CubicBezier
HitInfo
                                                 on click()
<u>Keys</u>
                                                 on double click()
<u>Light</u>
DirectionalLight
                                                   fb = FileBrowser(file_types=('.*'), enabled=True)
PointLight
<u>AmbientLight</u>
                                                   def on_submit(paths):
<u>SpotLight</u>
                                                        print('----'
                                                                         , paths)
<u>Ursina</u>
                                                        for p in paths:
<u>MeshModes</u>
                                                            print('---', p)
<u>Mesh</u>
<u>Wait</u>
                                                   fb.on_submit = on_submit
<u>Func</u>
<u>Sequence</u>
Shader
<u>Texture</u>
<u>Trigger</u>
                                               FileBrowser(<u>Entity</u>)
<u>Empty</u>
                                                 ursina.prefabs.file_browser
LoopingList
Vec2
                                                 FileBrowser(**kwargs)
Vec3
                                                 file types = ['.*']
                                                 file_types = ['.*', ]
start_path = Path('.').resolve()
Animation
Animator
                                                 return_files = True
<u>ButtonGroup</u>
                                                 return_folders = False
ButtonList
                                                 selection_limit = 1
Node
                                                 max buttons = 24
<u>Conversation</u>
                                                 title_bar = Button(parent=self, scale=(.9,.035), text='<gray>Open',
Cursor
                                                 color=color.dark_gray, collision=False)
DebugMenu
                                                 address_bar = Button(parent=self, scale=(.8,.035), text='//',
<u>Draggable</u>
<u>DropdownMenuButton</u>
                                                 text_origin=(-.5,0), y=-.05, highlight_color=color.black)
                                                 folder_up_button = Button(parent=self, scale=(.035,.035);
<u>DropdownMenu</u>
                                                 texture='arrow_down', rotation_z=180, position=(-.42,-.05),
<u>EditorCamera</u>
                                                 color=color.white, highlight_color=color.azure,
ExitButton
                                                 on_click=self.folder_up)
FileButton
                                                 button_parent = Entity(parent=self)
<u>FileBrowser</u>
                                                 back_panel = Entity(parent=self, model='quad', collider='box'
<u>FileBrowserSave</u>
                                                 origin_y=.5, scale=(.9,(self.max_buttons*.025)+.19), color=color._32,
<u>FirstPersonController</u>
                                                 z=.1)
FrameAnimation3d
                                                 bg = Button(parent=self, z=1, scale=(999,999), color=color.black66,
<u>GridEditor</u>
                                                 highlight color=color.black66, pressed color=color.black66)
<u>PixelEditor</u>
ASCIIEditor
                                                 cancel_button = Button(parent=self, scale=(.875*.24, .05), y=(-
                                                 self.max_buttons*.025)-.15, origin_x=-.5, x=-.875/2, text='Cancel',
HealthBar
<u>HotReloader</u>
```

```
open_button = Button(parent=self, scale=(.875*.74, .05), y=(-
                                                self.max_buttons*.025)-.15, origin_x=.5, x=.875/2, text='Open',
                                                color=color.dark_gray, on_click=self.open)
light dark
                                                cancel_button_2 = Button(parent=self.title_bar, model=Circle(),
                                                world_scale=self.title_bar.world_scale_y*.75, origin_x=.5, x=.495,
Entity
                                                z=-.1, text='<gray>x', on_click=self.close)
<u>Text</u>
Button
                                                can_scroll_up_indicator = Entity(parent=self, model='quad',
                                                texture='arrow_down', rotation_z=180, scale=(.05,.05), y=-.0765,
mouse
                                                z=-.1, color=color.dark_gray, enabled=False,
<u>raycaster</u>
                                                add to scene entities=False)
                                                can_scroll_down_indicator = Entity(parent=self, model='quad',
<u>application</u>
                                                texture='arrow_down', scale=(.05,.05), y=(-
build
camera
                                                self.max_buttons*.025)-.104, z=-.1, color=color.dark_gray,
color
                                                enabled=False, add to scene entities=False)
                                                scrol1
curve
<u>duplicate</u>
                                                path
                                                selection
input handler
<u>main</u>
                                                input(key)
mesh_importer
<u>scene</u>
                                                on enable()
shader
                                                close()
string_utilities
                                                folder_up()
                                                open(path=None)
text
<u>texture_importer</u>
                                                  fb = FileBrowser(file_types=('.*'), enabled=True)
<u>ursinamath</u>
ursinastuff
<u>window</u>
                                                   def on_submit(paths):
                                                                        ', paths)
                                                       print('----
                                                       for p in paths:
<u>Audio</u>
                                                           print('---', p)
Collider
BoxCollider
                                                  fb.on_submit = on_submit
<u>SphereCollider</u>
<u>MeshCollider</u>
<u>CollisionZone</u>
Color
CubicBezier
HitInfo
                                               FileBrowserSave(FileBrowser)
<u>Keys</u>
                                                ursina.prefabs.file browser save
<u>Light</u>
DirectionalLight
                                                FileBrowserSave(**kwargs)
PointLight
<u>AmbientLight</u>
                                                save button = self.open button
<u>SpotLight</u>
                                                file_name_field = InputField(
<u>Ursina</u>
                                                         parent
<u>MeshModes</u>
                                                file_type = '' # to save as
<u>Mesh</u>
<u>Wait</u>
                                                last_saved_file = None
                                                                             # gets set when you save a file
<u>Func</u>
                                                overwrite_prompt = WindowPanel(
<u>Sequence</u>
                                                         content=(
<u>Shader</u>
                                                              Text('Overwrite?'),
<u>Texture</u>
                                                              Button('Yes', color=color.azure, on_click=self.save),
<u>Trigger</u>
                                                              Button('Cancel')
<u>Empty</u>
                                                         ),
LoopingList
                                                         z=-1,
Vec2
                                                         popup=True,
<u>Vec3</u>
                                                         enabled=False)
Animation
                                                save()
Animator
ButtonGroup
                                                  from ursina.prefabs.file_browser_save import FileBrowserSave
ButtonList
Node
                                                  wp = FileBrowserSave(file type = '.oto')
<u>Conversation</u>
Cursor
DebugMenu
                                                   import json
<u>Draggable</u>
                                                   save data = {'level': 4, 'name':'Link'}
<u>DropdownMenuButton</u>
                                                  wp.data = json.dumps(save_data)
<u>DropdownMenu</u>
<u>EditorCamera</u>
ExitButton
FileButton
<u>FileBrowser</u>
                                               FirstPersonController(<u>Entity</u>)
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
                                                ursina.prefabs.first_person_controller
FrameAnimation3d
<u>GridEditor</u>
                                                FirstPersonController(**kwargs)
<u>PixelEditor</u>
ASCIIEditor
                                                cursor = Entity(parent=camera.ui, model='quad', color=color.pink,
```

scale=.008, rotation z=45)

<u>HealthBar</u>

<u>HotReloader</u>

on\_click=self.close)

```
height = 2
                                                 camera_pivot = Entity(parent=self, y=self.height)
light dark
                                                 mouse sensitivity = Vec2(40, 40)
                                                 gravity = 1
Entity
                                                 grounded = False
<u>Text</u>
                                                 jump\_height = 2
Button
                                                 jump_up_duration = .5
                                                 fall_after = .35 # will interrupt jump up
mouse
                                                 jumping = False
<u>raycaster</u>
                                                 air_time = 0
<u>application</u>
                                                 update()
build
camera
                                                 input(key)
color
                                                 jump()
                                                 start_fall()
curve
<u>duplicate</u>
                                                 land()
input handler
                                                 on enable()
                                                 on disable()
<u>main</u>
mesh_importer
<u>scene</u>
                                                    from ursina.prefabs.first_person_controller import
shader
                                                   FirstPersonController
                                                   ground = Entity(model='plane', scale=(100,1,100),
string_utilities
                                                   color=color.yellow.tint(-.2), texture='white_cube', texture_scale=
text
<u>texture_importer</u>
                                                   (100,100), collider='box')
                                                   e = Entity(model='cube', scale=(1,5,10), x=2, y=.01, rotation_y=45,
<u>ursinamath</u>
                                                   collider='box', texture='white_cube')
ursinastuff
                                                   e.texture_scale = (e.scale_z, e.scale_y)
<u>window</u>
                                                   e = Entity(model='cube', scale=(1,5,10), x=-2, y=.01, collider='box',
                                                   texture='white_cube')
<u>Audio</u>
Collider
                                                   e.texture_scale = (e.scale_z, e.scale_y)
BoxCollider
                                                   player = FirstPersonController(y=2, origin_y=-.5)
<u>SphereCollider</u>
                                                   player.gun = None
<u>MeshCollider</u>
<u>CollisionZone</u>
Color
CubicBezier
                                                   gun = Button(parent=scene, model='cube', color=color.blue,
HitInfo
                                                   origin_y=-.5, position=(3,0,3), collider='box')
                                                   gun.on click = Sequence(Func(setattr, gun, 'parent', camera),
<u>Keys</u>
<u>Light</u>
                                                   Func(setattr, player, 'gun', gun))
DirectionalLight
                                                   gun_2 = duplicate(gun, z=7, x=8)
PointLight
                                                   slope = Entity(model='cube', collider='box', position=(0,0,8),
scale=6, rotation=(45,0,0), texture='brick', texture_scale=(8,8))
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
                                                   slope = Entity(model='cube', collider='box', position=(5,0,10),
<u>MeshModes</u>
                                                   scale=6, rotation=(80,0,0), texture='brick', texture_scale=(8,8))
<u>Mesh</u>
<u>Wait</u>
                                                   hookshot_target = Button(parent=scene, model='cube',
<u>Func</u>
                                                    color=color.brown, position=(4,5,5))
                                                   hookshot_target.on_click = Func(player.animate_position,
<u>Sequence</u>
                                                   hookshot_target.position, duration=.5, curve=curve.linear)
<u>Shader</u>
<u>Texture</u>
<u>Trigger</u>
                                                   def input(key):
                                                        if key == 'left mouse down' and player.gun:
<u>Empty</u>
                                                             gun.blink(color.orange)
LoopingList
                                                             bullet = Entity(parent=gun, model='cube', scale=.1,
Vec2
                                                   color=color.black)
Vec3
                                                             bullet.world_parent = scene
                                                             bullet.animate position(bullet.position+(bullet.forward*50),
Animation
Animator
                                                    curve=curve.linear, duration=1)
<u>ButtonGroup</u>
                                                             destroy(bullet, delay=1)
ButtonList
Node
<u>Conversation</u>
Cursor
DebugMenu
                                               FrameAnimation3d(<u>Entity</u>)
Draggable
                                                 ursina.prefabs.frame_animation_3d
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
                                                 FrameAnimation3d(name, fps=12, loop=True, autoplay=True,
<u>EditorCamera</u>
                                                 frame_times=None, **kwargs)
ExitButton
FileButton
                                                 frames = [Entity(parent=self, model=e.stem, enabled=False,
<u>FileBrowser</u>
                                                 add to scene entities=False) for e in model names]
<u>FileBrowserSave</u>
                                                 sequence = Sequence(loop=loop, auto_destroy=False)
<u>FirstPersonController</u>
                                                 is_playing = False
FrameAnimation3d
                                                 autoplay = autoplay
<u>GridEditor</u>
                                                 duration
<u>PixelEditor</u>
ASCIIEditor
                                                 start()
<u>HealthBar</u>
                                                 pause()
```

<u>HotReloader</u>

speed = 5

```
resume()
                                                 finish()
light dark
                                                   application.asset folder = application.asset folder.parent.parent /
                                                    'samples'
Entity
<u>Text</u>
Button
                                                   Loads an obj sequence as a frame animation.
                                                   So if you have some frames named run_cycle_000.obj,
mouse
                                                   run_cycle_001.obj, run_cycle_002.obj and so on,
<u>raycaster</u>
                                                   you can load it like this: FrameAnimation3d('run_cycle_')
<u>application</u>
build
camera
                                                   FrameAnimation3d('blob animation ')
color
curve
<u>duplicate</u>
input handler
                                               GridEditor(Entity)
<u>main</u>
mesh_importer
                                                 ursina.prefabs.grid_editor
<u>scene</u>
shader
                                                 GridEditor(size=(32,32), palette=(' ', '#', '|', 'o'), **kwargs)
string_utilities
text
                                                 w, self.h = int(size[0]), int(size[1])
<u>texture_importer</u>
                                                 brush\_size = 1
<u>ursinamath</u>
                                                 auto render = True
ursinastuff
                                                 cursor = Entity(parent=self, model=Quad(segments=0, mode='line'),
<u>window</u>
                                                 origin=(-.5,-.5), scale=(1/self.w, 1/self.h),
                                                 color=color.color(0,1,1,.5), z=-.1)
<u>Audio</u>
                                                 selected char = palette[1]
Collider
                                                 palette = palette
BoxCollider
                                                 prev_draw = None
<u>SphereCollider</u>
                                                 start_pos = (0,0)
<u>MeshCollider</u>
                                                 outline = Entity(parent=self, model=Quad(segments=0, mode='line',
<u>CollisionZone</u>
                                                 thickness=1), color=color.cyan, z=.01, origin=(-.5,-.5))
Color
                                                 undo_cache = list()
CubicBezier
                                                 undo_index = 0
HitInfo
                                                 help_text = Text(
<u>Keys</u>
                                                          text=dedent('''
<u>Light</u>
                                                               left mouse:
                                                                               draw
<u>DirectionalLight</u>
                                                               control(hold): draw lines
<u>PointLight</u>
                                                               alt(hold): select character
<u>AmbientLight</u>
                                                               ctrl + z:
                                                                               undo
<u>SpotLight</u>
                                                               ctrl + y:
                                                                                redo
<u>Ursina</u>
<u>MeshModes</u>
                                                          position=window.top left,
<u>Mesh</u>
                                                          scale=.75
<u>Wait</u>
<u>Func</u>
                                                 edit mode = True
<u>Sequence</u>
Shader
                                                 update()
<u>Texture</u>
                                                 draw(x, y)
<u>Trigger</u>
                                                 input(key)
<u>Empty</u>
                                                 record_undo()
LoopingList
                                                 floodfill(matrix, x, y, first=True)
Vec2
Vec3
                                                   pixel editor example, it's basically a drawing tool.
Animation
                                                   can be useful for level editors and such
Animator
                                                   here we create a new texture, but can also give it an existing
<u>ButtonGroup</u>
                                                   texture to modify.
ButtonList
Node
                                                   from PIL import Image
<u>Conversation</u>
                                                   t = Texture(Image.new(mode='RGBA', size=(32,32), color=(0,0,0,1)))
Cursor
                                                   from ursina.prefabs.grid_editor import PixelEditor
DebugMenu
                                                   PixelEditor(texture=load_texture('brick'))
<u>Draggable</u>
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
                                                   same as the pixel editor, but with text.
<u>EditorCamera</u>
ExitButton
                                                   from ursina.prefabs.grid editor import ASCIIEditor
FileButton
                                                   ASCIIEditor()
<u>FileBrowser</u>
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
FrameAnimation3d
```

## PixelEditor(GridEditor)

ursina.prefabs.grid editor

<u>GridEditor</u>

<u>PixelEditor</u> ASCIIEditor

<u>HealthBar</u> <u>HotReloader</u>

```
color.light_gray, color.gray, color.red, color.orange, color.yellow,
                                                color.lime, color.green, color.turquoise, color.cyan, color.azure,
<u>light</u> <u>dark</u>
                                                color.blue, color.violet, color.magenta, color.pink), **kwargs)
Entity
                                                grid = [[texture.get_pixel(x,y) for y in range(texture.height)] for x
                                                in range(texture.width)]
<u>Text</u>
Button
                                                render()
mouse
                                                save()
<u>raycaster</u>
<u>application</u>
                                                  pixel editor example, it's basically a drawing tool.
build
camera
                                                  can be useful for level editors and such
color
                                                  here we create a new texture, but can also give it an existing
curve
                                                  texture to modify.
<u>duplicate</u>
input handler
                                                  from PIL import Image
                                                  t = Texture(Image.new(mode='RGBA', size=(32,32), color=(0,0,0,1)))
<u>main</u>
                                                  from ursina.prefabs.grid_editor import PixelEditor
mesh_importer
<u>scene</u>
                                                  PixelEditor(texture=load_texture('brick'))
shader
                                                   1.1.1
string_utilities
                                                  same as the pixel editor, but with text.
text
<u>texture_importer</u>
<u>ursinamath</u>
                                                  from ursina.prefabs.grid_editor import ASCIIEditor
ursinastuff
                                                  ASCIIEditor()
<u>window</u>
<u>Audio</u>
Collider
BoxCollider
                                              ASCIIEditor(GridEditor)
<u>SphereCollider</u>
                                                ursina.prefabs.grid_editor
<u>MeshCollider</u>
<u>CollisionZone</u>
                                                ASCIIEditor(size=(61,28), palette=(' ', '#', '|', 'A', '/', '\\', 'o',
Color
                                                  _', '-', 'i', 'M', '.'), font='VeraMono.ttf', color=color.black,
CubicBezier
                                                line_height=1.1, **kwargs)
HitInfo
<u>Keys</u>
                                                text_entity = Text(parent=self.parent, text=text, x=self.x,
<u>Light</u>
                                                line_height=line_height, font=font)
DirectionalLight
                                                scale = (self.text_entity.width, self.text_entity.height)
PointLight
<u>AmbientLight</u>
                                                render()
<u>SpotLight</u>
                                                input(key)
<u>Ursina</u>
<u>MeshModes</u>
                                                 1.1.1
<u>Mesh</u>
                                                  pixel editor example, it's basically a drawing tool.
<u>Wait</u>
                                                  can be useful for level editors and such
<u>Func</u>
                                                  here we create a new texture, but can also give it an existing
<u>Sequence</u>
                                                  texture to modify.
Shader
<u>Texture</u>
                                                  from PIL import Image
<u>Trigger</u>
                                                  t = Texture(Image.new(mode='RGBA', size=(32,32), color=(0,0,0,1)))
<u>Empty</u>
                                                  from ursina.prefabs.grid_editor import PixelEditor
LoopingList
                                                  PixelEditor(texture=load_texture('brick'))
Vec2
<u>Vec3</u>
                                                  same as the pixel editor, but with text.
Animation
Animator
                                                  from ursina.prefabs.grid editor import ASCIIEditor
ButtonGroup
                                                  ASCIIEditor()
ButtonList
Node
<u>Conversation</u>
Cursor
DebugMenu
                                              HealthBar(<u>Button</u>)
<u>Draggable</u>
<u>DropdownMenuButton</u>
                                                ursina.prefabs.health_bar
<u>DropdownMenu</u>
EditorCamera
                                                HealthBar(max_value=100, show_text=True, show_lines=False, **kwargs)
ExitButton
FileButton
                                                bar = Entity(parent=self, model='quad', origin=self.origin, z=-.01,
<u>FileBrowser</u>
                                                color=color.red.tint(-.2), ignore=True)
<u>FileBrowserSave</u>
                                                animation_duration = .1
<u>FirstPersonController</u>
                                                lines = Entity(parent=self.bar, origin=self.origin, y=-1,
FrameAnimation3d
                                                color=color.black33, ignore=True)
<u>GridEditor</u>
                                                roundness = .25
<u>PixelEditor</u>
                                                max_value = max_value
ASCIIEditor
                                                clamp = True
HealthBar
                                                show lines = show lines
<u>HotReloader</u>
```

PixelEditor(texture, palette=(color.black, color.white,

```
scale_x = self.scale_x # update rounded corners
                                                    scale_y = self.scale_y # update background's rounded corners
light dark
                                                    value = self.max value
Entity
<u>Text</u>
Button
                                                       roundness=.5, value=50)
mouse
<u>raycaster</u>
<u>application</u>
build
camera
                                                                health bar 1.value -= 10
color
curve
<u>duplicate</u>
input handler
                                                  HotReloader(Entity)
<u>main</u>
mesh_importer
                                                    ursina.prefabs.hot reloader
<u>scene</u>
shader
                                                    HotReloader(path=__file__, **kwargs)
string_utilities
text
                                                    path = path
<u>texture_importer</u>
                                                    path = Path(self.path)
<u>ursinamath</u>
ursinastuff
<u>window</u>
                                                              'ctrl+r' : self.reload_code,
                                                                        : self.reload_code,
<u>Audio</u>
                                                              'f6'
Collider
                                                              'f7'
'f8'
BoxCollider
<u>SphereCollider</u>
<u>MeshCollider</u>
<u>CollisionZone</u>
Color
                                                    input(key)
CubicBezier
                                                    update()
HitInfo
                                                    toggle_hotreloading()
<u>Keys</u>
                                                    hot_reload()
<u>Light</u>
                                                    reload_code(reset_camera=True)
DirectionalLight
                                                    reload_textures()
PointLight
                                                    reload_models()
<u>AmbientLight</u>
                                                    reload_shaders()
<u>SpotLight</u>
<u>Ursina</u>
                                                       application.hot reloader.path =
<u>MeshModes</u>
<u>Mesh</u>
<u>Wait</u>
<u>Func</u>
<u>Sequence</u>
Shader
<u>Texture</u>
<u>Trigger</u>
<u>Empty</u>
LoopingList
Vec2
                                                  ContentTypes
Vec3
                                                    ursina.prefabs.input_field
Animation
Animator
                                                    ContentTypes(path=__file__, **kwargs)
ButtonGroup
ButtonList
Node
<u>Conversation</u>
Cursor
DebugMenu
<u>Draggable</u>
<u>DropdownMenuButton</u>
                                                      color=color.hsv(240,.6,.1,.75))
<u>DropdownMenu</u>
EditorCamera
ExitButton
FileButton
<u>FileBrowser</u>
<u>FileBrowserSave</u>
                                                      def submit():
<u>FirstPersonController</u>
FrameAnimation3d
<u>GridEditor</u>
<u>PixelEditor</u>
ASCIIEditor
```

HealthBar <u>HotReloader</u> show\_text = show\_text

```
health_bar_1 = HealthBar(bar_color=color.lime.tint(-.25),
  def input(key):
    if key == '+' or key == '+ hold':
      health_bar_1.value += 10 if key == '-' or key == '- hold':
realtime editing = False # toggle with f8
                   : self.reload textures,
         'f8' : self.reload_shaders,
# 'f9' : self.reload_shaders,
                   : self.reload models,
                    : self.toggle_hotreloading,
  application.asset_folder.parent.parent / 'samples' / 'platformer.py'
  By default you can press F5 to reload the starting script, F6 to
  reimport textures and F7 to reload models.
```

```
background = Entity(model='quad', texture='pixelscape_combo',
parent=camera.ui, scale=(camera.aspect_ratio,1), color=color.white)
gradient = Entity(model='quad', texture='vertical_gradient',
parent=camera.ui, scale=(camera.aspect_ratio,1),
username_field = InputField(y=-.12, limit_content_to='0123456789')
password_field = InputField(y=-.18, hide_content=True)
username_field.next_field = password_field
    print('ursername:', username_field.text)
    print('password:', password_field.text)
Button('Login', scale=.1, color=color.cyan.tint(-.4), y=-.26,
on click=submit).fit to text()
```

```
light dark
Entity
<u>Text</u>
Button
mouse
<u>raycaster</u>
<u>application</u>
build
camera
color
curve
<u>duplicate</u>
input handler
<u>main</u>
mesh_importer
<u>scene</u>
<u>shader</u>
string_utilities
text
<u>texture_importer</u>
<u>ursinamath</u>
ursinastuff
<u>window</u>
<u>Audio</u>
Collider
BoxCollider
<u>SphereCollider</u>
<u>MeshCollider</u>
<u>CollisionZone</u>
Color
CubicBezier
HitInfo
<u>Keys</u>
<u>Light</u>
<u>DirectionalLight</u>
PointLight
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
<u>Mesh</u>
<u>Wait</u>
<u>Func</u>
<u>Sequence</u>
Shader
<u>Texture</u>
<u>Trigger</u>
<u>Empty</u>
LoopingList
Vec2
<u>Vec3</u>
Animation
Animator
<u>ButtonGroup</u>
ButtonList
<u>Node</u>
Conversation
Cursor
DebugMenu
<u>Draggable</u>
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
<u>EditorCamera</u>
<u>ExitButton</u>
FileButton
<u>FileBrowser</u>
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
FrameAnimation3d
<u>GridEditor</u>
<u>PixelEditor</u>
ASCIIEditor
HealthBar
```

```
InputField(Button)
 ursina.prefabs.input field
```

```
InputField(default_value='', label='', max_lines=1, max_width=24,
  **kwargs)
 default_value = default_value
 limit_content_to = None
                        # if set to True, will display content as '*'.
 hide content = False
 can also be set to character instead of True.
 next_field = None
 text field = TextField(world parent=self, x=-.45, y=.3, z=-.1,
 max_lines=max_lines)
 active = False
 text_color
 render()
 input(key)
   background = Entity(model='quad', texture='pixelscape_combo',
   parent=camera.ui, scale=(camera.aspect_ratio,1), color=color.white)
   gradient = Entity(model='quad', texture='vertical_gradient',
   parent=camera.ui, scale=(camera.aspect_ratio,1),
   color=color.hsv(240,.6,.1,.75))
   username_field = InputField(y=-.12, limit_content_to='0123456789')
   password_field = InputField(y=-.18, hide_content=True)
   username_field.next_field = password_field
   def submit():
        print('ursername:', username_field.text)
print('password:', password_field.text)
   Button('Login', scale=.1, color=color.cyan.tint(-.4), y=-.26,
   on click=submit).fit to text()
MemoryCounter(Text)
 ursina.prefabs.memory_counter
 MemoryCounter(**kwargs)
 parent = camera.ui
 position = window.bottom right - Vec2(.025,0)
 origin = (0.5, -0.5)
 process = psutil.Process(os.getpid())
 i = 0
 text = 'eofiwjeofiwejf'
 update()
   MemoryCounter()
   Displays the amount of memory used in the bottom right corner
Panel(<u>Entity</u>)
```

# ursina.prefabs.panel

```
Panel(**kwargs)
```

```
parent = camera.ui
model = Quad()
color = Button.color
```

```
p = Panel()
```

```
light dark
Entity
<u>Text</u>
Button
mouse
<u>raycaster</u>
<u>application</u>
build
camera
color
curve
<u>duplicate</u>
input handler
<u>main</u>
mesh_importer
<u>scene</u>
shader
string_utilities
text
<u>texture_importer</u>
<u>ursinamath</u>
ursinastuff
<u>window</u>
<u>Audio</u>
Collider
BoxCollider
<u>SphereCollider</u>
<u>MeshCollider</u>
<u>CollisionZone</u>
Color
CubicBezier
HitInfo
<u>Keys</u>
<u>Light</u>
DirectionalLight
PointLight
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
<u>Mesh</u>
<u>Wait</u>
<u>Func</u>
<u>Sequence</u>
<u>Shader</u>
<u>Texture</u>
<u>Trigger</u>
<u>Empty</u>
LoopingList
Vec2
<u>Vec3</u>
Animation
Animator
ButtonGroup
ButtonList
Node
Conversation
Cursor
DebugMenu
<u>Draggable</u>
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
<u>EditorCamera</u>
ExitButton
FileButton
<u>FileBrowser</u>
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
FrameAnimation3d
<u>GridEditor</u>
<u>PixelEditor</u>
ASCIIEditor
HealthBar
```

## PlatformerController2d(<u>Entity</u>)

ursina.prefabs.platformer\_controller\_2d

```
PlatformerController2d(**kwargs)
```

```
model = 'cube'
origin_y = -.5
scale_y = 2
color = color.orange
collider = 'box'
animator = Animator({'idle' : None, 'walk' : None, 'jump' : None})
walk speed = 8
walking = False
velocity = 0 # the walk diection is stored here. -1 for left and 1 for
jump\_height = 4
jump_duration = .5
jumping = False
max_jumps = 1
jumps_left = self.max_jumps
gravity = 1
grounded = True
air_time = 0  # this increase while we're falling and used when
calculating the distance we fall so we fall faster and faster instead
of linearly.
traverse_target = scene
                           # by default, it will collide with
everything except itself. you can change this to change the boxcast
traverse target.
gravity = 0
update()
input(key)
jump()
start fall()
land()
  camera.orthographic = True
  camera.fov = 10
  ground = Entity(model='cube', color=color.white33, origin_y=.5,
  scale=(20, 10, 1), collider='box')
  wall = Entity(model='cube', color=color.azure, origin=(-.5,.5),
  scale=(5,10), x=10, y=.5, collider='box')
  wall_2 = Entity(model='cube', color=color.white33, origin=(-.5,.5),
  scale=(5,10), x=10, y=5, collider='box')
ceiling = Entity(model='cube', color=color.white33, origin_y=-.5,
  scale=(1, 1, 1), y=1, collider='box')
  def input(key):
      if key == 'c':
          wall.collision = not wall.collision
          print(wall.collision)
```

player controller = PlatformerController2d(scale y=2, jump height=4,

camera.add\_script(SmoothFollow(target=player\_controller, offset=

### RadialMenu(Entity)

[0,1,-30], speed=4))

EditorCamera()

ursina.prefabs.radial\_menu

#### RadialMenu(buttons=list(), \*\*kwargs)

```
color=color.color(0,0,0,.1),
                                                               enabled=False)
light dark
                                                     z = -99
                                                     scale = .075
Entity
                                                     on enable()
<u>Text</u>
Button
                                                     input(key)
mouse
                                                       rm = RadialMenu(
<u>raycaster</u>
                                                            buttons = (
                                                                 RadialMenuButton(text='1'),
<u>application</u>
                                                                 RadialMenuButton(text='2'),
build
camera
                                                                 RadialMenuButton(text='3'),
                                                                 RadialMenuButton(text='4'),
RadialMenuButton(text='5', scale=.5),
RadialMenuButton(text='6', color=color.red),
color
curve
<u>duplicate</u>
input handler
                                                            enabled = False
<u>main</u>
mesh_importer
<u>scene</u>
                                                       RadialMenuButton(text='6', color=color.red,x =-.5, scale=.06),
shader
                                                       def enable_radial_menu():
                                                            rm.enabled = True
string_utilities
                                                       cube = Button(parent=scene, model='cube', color=color.orange,
text
<u>texture_importer</u>
                                                       highlight_color=color.azure, on_click=enable_radial_menu)
<u>ursinamath</u>
                                                       EditorCamera()
ursinastuff
<u>window</u>
<u>Audio</u>
Collider
                                                   RadialMenuButton(<u>Button</u>)
<u>BoxCollider</u>
                                                     ursina.prefabs.radial menu
<u>SphereCollider</u>
<u>MeshCollider</u>
                                                     RadialMenuButton(**kwargs)
<u>CollisionZone</u>
Color
CubicBezier
<u>HitInfo</u>
                                                       rm = RadialMenu(
<u>Keys</u>
                                                            buttons = (
<u>Light</u>
                                                                 RadialMenuButton(text='1'),
DirectionalLight
                                                                 RadialMenuButton(text='2'),
PointLight
                                                                 RadialMenuButton(text='3'),
<u>AmbientLight</u>
                                                                 RadialMenuButton(text='4'),
<u>SpotLight</u>
                                                                 RadialMenuButton(text='5', scale=.5),
RadialMenuButton(text='6', color=color.red),
<u>Ursina</u>
<u>MeshModes</u>
<u>Mesh</u>
                                                            enabled = False
<u>Wait</u>
<u>Func</u>
                                                       RadialMenuButton(text='6', color=color.red,x =-.5, scale=.06),
<u>Sequence</u>
                                                       def enable radial menu():
<u>Shader</u>
                                                            rm.enabled = True
<u>Texture</u>
                                                       cube = Button(parent=scene, model='cube', color=color.orange,
<u>Trigger</u>
                                                       highlight color=color.azure, on click=enable radial menu)
<u>Empty</u>
                                                       EditorCamera()
<u>LoopingList</u>
Vec2
Vec3
Animation
                                                   Sky(Entity)
Animator
ButtonGroup
                                                     ursina.prefabs.sky
ButtonList
Node
                                                     Sky(**kwargs)
Conversation
Cursor
DebugMenu
                                                     update()
<u>Draggable</u>
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
<u>EditorCamera</u>
<u>ExitButton</u>
                                                   Slider(<u>Entity</u>)
FileButton
                                                     ursina.prefabs.slider
<u>FileBrowser</u>
<u>FileBrowserSave</u>
                                                     Slider(min=0, max=1, default=None, height=Text.size, text='',
<u>FirstPersonController</u>
                                                     dynamic=False, **kwargs)
FrameAnimation3d
<u>GridEditor</u>
                                                     parent = camera.ui
<u>PixelEditor</u>
ASCIIEditor
                                                     vertical = False
                                                     min = min
<u>HealthBar</u>
<u>HotReloader</u>
```

collider='box',

```
default = default
                                                 step = 0
light dark
                                                 height = height
                                                 on_value_changed = None
                                                                              # set this to a function you want to be
Entity
                                                 called when the slider changes
                                                                               # set this to (object, 'attrname') to set
<u>Text</u>
                                                 setattr = None
Button
                                                 that value when the slider changes
                                                 label = Text(parent=self, origin=(0.5, 0), x=-0.025, text=text)
mouse
                                                 bg = Entity(parent=self, model=Quad(scale=(.525, height),
<u>raycaster</u>
                                                 radius=Text.size/2, segments=3),
                                                          origin_x=-0.25, collider='box', color=color.black66)
<u>application</u>
                                                 knob = Draggable(parent=self, min_x=0, max_x=.5, min_y=0, max_y=.5,
build
camera
                                                 step=self.step.
color
                                                          model=Quad(radius=Text.size/2, scale=(Text.size, height)),
                                                 collider='box', color=color.light_gray,
curve
                                                         text='0', text_origin=(0, -.55), z=-.1)
<u>duplicate</u>
input handler
                                                 value = self.default
                                                                       # if set to True, will call on_value_changed()
                                                 dynamic = dynamic
<u>main</u>
                                                 while dragging. if set to False, will only call on_value_changed()
mesh_importer
<u>scene</u>
                                                 after dragging.
shader
string_utilities
                                                 bg_click()
                                                 drop()
text
<u>texture_importer</u>
                                                 update()
ursinamath
                                                 slide()
<u>ursinastuff</u>
                                                   box = Entity(model='cube', origin_y=-.5, scale=1, color=color.orange)
<u>window</u>
<u>Audio</u>
                                                   def scale_box():
                                                       box.scale_y = slider.value
Collider
BoxCollider
                                                       print(thin_slider.value)
<u>SphereCollider</u>
<u>MeshCollider</u>
                                                   slider = Slider(0, 20, default=10, height=Text.size*3, y=-.4, step=1,
<u>CollisionZone</u>
                                                   on_value_changed=scale_box, vertical=True)
Color
CubicBezier
                                                   thin_slider = ThinSlider(text='height', dynamic=True,
<u>HitInfo</u>
                                                   on value changed=scale box)
<u>Keys</u>
<u>Light</u>
                                                   thin_slider.label.origin = (∅,∅)
                                                   thin_slider.label.position = (.25, -.1)
<u>DirectionalLight</u>
PointLight
<u>AmbientLight</u>
SpotLight
<u>Ursina</u>
<u>MeshModes</u>
                                               ThinSlider(Slider)
<u>Mesh</u>
                                                 ursina.prefabs.slider
<u>Wait</u>
<u>Func</u>
                                                 ThinSlider(*args, **kwargs)
<u>Sequence</u>
Shader
<u>Texture</u>
<u>Trigger</u>
                                                   box = Entity(model='cube', origin y=-.5, scale=1, color=color.orange)
<u>Empty</u>
LoopingList
                                                   def scale_box():
Vec2
                                                       box.scale_y = slider.value
<u>Vec3</u>
                                                       print(thin_slider.value)
Animation
                                                   slider = Slider(0, 20, default=10, height=Text.size*3, y=-.4, step=1,
Animator
                                                   on_value_changed=scale_box, vertical=True)
<u>ButtonGroup</u>
ButtonList
                                                   thin_slider = ThinSlider(text='height', dynamic=True,
<u>Node</u>
                                                   on_value_changed=scale_box)
Conversation
Cursor
                                                   thin_slider.label.origin = (0,0)
DebugMenu
                                                   thin_slider.label.position = (.25, -.1)
<u>Draggable</u>
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
<u>EditorCamera</u>
<u>ExitButton</u>
                                               Sprite(Entity)
FileButton
<u>FileBrowser</u>
                                                 ursina.prefabs.sprite
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
                                                 Sprite(texture=None, ppu=ppu, **kwargs)
FrameAnimation3d
<u>GridEditor</u>
                                                 model = 'quad'
<u>PixelEditor</u>
                                                 texture = texture
ASCIIEditor
                                                 ppu = ppu
<u>HealthBar</u>
                                                 aspect ratio = self.texture.width / self.texture.height
<u>HotReloader</u>
```

max = max

```
scale_x = self.scale_y * self.aspect_ratio
```

```
light dark
                                                       camera.orthographic = True
                                                       camera.fov = 1
Entity
                                                       Sprite.ppu = 16
<u>Text</u>
                                                       Texture.default_filtering = None
Button
                                                       s = Sprite('brick', filtering=False)
mouse
<u>raycaster</u>
<u>application</u>
<u>build</u>
                                                   TextField(Entity)
camera
                                                     ursina.prefabs.text_field
color
curve
                                                     TextField(**kwargs)
<u>duplicate</u>
input handler
                                                     font = 'VeraMono.ttf'
                                                     line_height = 1
mesh_importer
                                                     max lines = 99999
<u>scene</u>
                                                     text_entity = Text(
shader
                                                              parent
string_utilities
                                                     line numbers = Text(
text
<u>texture_importer</u>
                                                     character_width = Text.get_width('a', font=self.font)
<u>ursinamath</u>
                                                     cursor parent = Entity(parent=self, scale=(self.character width,
ursinastuff
<u>window</u>
                                                     cursor = Entity(parent=self.cursor_parent, model='cube',
                                                     color=color.white33, origin=(-.5, -.5), scale=(.1, 1))
<u>Audio</u>
                                                     bg = Entity(parent=self.cursor_parent, model='cube',
color=color.dark_gray, origin=(-.5,-.5), z=1, scale=(120, 20),
Collider
BoxCollider
                                                     collider='box', visible=False)
<u>SphereCollider</u>
                                                     selection = None
<u>MeshCollider</u>
                                                     selection_parent = Entity(parent=self.cursor_parent)
<u>CollisionZone</u>
                                                     register_mouse_input = False
Color
CubicBezier
                                                     replacements = dict()
HitInfo
                                                     on_undo = list()
<u>Keys</u>
                                                     on redo = list()
<u>Light</u>
                                                     shifted_keys = {
<u>DirectionalLight</u>
PointLight
<u>AmbientLight</u>
SpotLight
<u>Ursina</u>
                                                               '<' : '>'
                                                               '<' : '>',
'+' : '?',
<u>MeshModes</u>
<u>Mesh</u>
                                                               '0' : '='
Wait
                                                               '1' : '!'
<u>Func</u>
                                                               '2' : '"'
<u>Sequence</u>
                                                               '3' : '#'
<u>Shader</u>
                                                               # '4' : '¤',
<u>Texture</u>
                                                               '5' : '%',
<u>Trigger</u>
                                                               '6' : '&',
<u>Empty</u>
                                                               '7': '/',
<u>LoopingList</u>
                                                               '8': '('
Vec2
                                                               '9': ')',
Vec3
                                                     alted_keys = {
Animation
                                                               '0' : '}'
                                                               '2': '@',
Animator
ButtonGroup
                                                               '3' : '£',
ButtonList
                                                               '4' : '¤',
'5' : '€',
'7' : '{',
Node
<u>Conversation</u>
Cursor
DebugMenu
<u>Draggable</u>
                                                     shortcuts = {
<u>DropdownMenuButton</u>
                                                                                      ('enter', 'enter hold'),
                                                               'newline':
<u>DropdownMenu</u>
                                                               'erase':
                                                                                       ('backspace', 'backspace hold'),
EditorCamera
                                                                                       ('ctrl+backspace', 'ctrl+backspace hold'),
('ctrl+shift+k',),
('ctrl+z', 'ctrl+z hold'),
('ctrl+y', 'ctrl+y hold', 'ctrl+shift+z',
                                                               'erase word':
ExitButton
                                                               'delete_line':
FileButton
                                                               'undo':
<u>FileBrowser</u>
                                                               'redo':
<u>FileBrowserSave</u>
                                                     'ctrl+shift+z hold'),
<u>FirstPersonController</u>
                                                              # 'save':
                                                                                          ('ctrl+s',),
FrameAnimation3d
                                                               # 'save_as':
                                                                                         ('ctrl+shift+s',),
<u>GridEditor</u>
                                                               'indent':
'dedent':
                                                                                       ('tab',),
PixelEditor
                                                                                       ('shift+tab',),
ASCIIEditor
                                                               'move_line_down': ('ctrl+down arrow', 'ctrl+down arrow
HealthBar
```

hold'),

<u>HotReloader</u>

```
# 'cut':
                                                                                    ('ctrl+x',),
                                                                                  ('ctrl+c',),
('ctrl+v',),
                                                           'copy':
<u>light</u> <u>dark</u>
                                                           'paste':
                                                           # 'select_all':
                                                                                   ('ctrl+a',),
                                                           # 'toggle_comment':
Entity
                                                                                  ('ctrl+alt+c',),
                                                           # 'find':
                                                                                    ('ctrl+f',),
<u>Text</u>
Button
                                                  blink_cursor()
mouse
<u>raycaster</u>
                                                  add_text(s, move_cursor=True)
                                                  move_line(a, b)
<u>application</u>
                                                  erase()
                                                  delete_selected()
build
                                                  input(key)
camera
color
                                                  render()
curve
                                                  update()
<u>duplicate</u>
                                                  select all()
input handler
                                                  draw selection()
<u>main</u>
                                                    window.x = 200
mesh_importer
<u>scene</u>
shader
                                                    window.color = color.color(0, 0, .1)
string_utilities
                                                    Button.color = color._20
                                                    window.color = color._25
text
<u>texture_importer</u>
                                                    Text.default_font = 'consola.ttf'
<u>ursinamath</u>
ursinastuff
                                                    Text.default_resolution = 16*2
<u>window</u>
                                                    te = TextField(max_lines=300, scale=1)
                                                    te.text = dedent(
<u>Audio</u>
                                                        Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Collider
                                                        Aliquam sapien tellus, venenatis sit amet ante et, malesuada
BoxCollider
<u>SphereCollider</u>
                                                        Etiam et mi luctus, viverra urna at, maximus eros. Sed dictum
                                                    faucibus purus,
<u>MeshCollider</u>
<u>CollisionZone</u>
                                                        nec rutrum ipsum condimentum in. Mauris iaculis arcu nec justo
Color
                                                    rutrum euismod.
CubicBezier
                                                        Suspendisse dolor tortor, congue id erat sit amet, sollicitudin
                                                    facilisis velit.'''
<u>HitInfo</u>
<u>Keys</u>
                                                        )[1:]
<u>Light</u>
                                                    te.render()
<u>DirectionalLight</u>
PointLight
<u>AmbientLight</u>
SpotLight
<u>Ursina</u>
                                                Tilemap(GridEditor)
<u>MeshModes</u>
                                                  ursina.prefabs.tilemap
<u>Mesh</u>
<u>Wait</u>
                                                  Tilemap(tilemap='', tileset='', tileset_size=(8,8), **kwargs)
<u>Func</u>
<u>Sequence</u>
                                                  grid = [[self.tilemap.get_pixel(x,y) for y in
Shader
                                                  range(self.tilemap.height)] for x in range(self.tilemap.width)]
<u>Texture</u>
                                                  tileset = tileset
<u>Trigger</u>
                                                 tileset size = tileset size
<u>Empty</u>
                                                  model = Mesh()
LoopingList
                                                  texture = tileset
Vec2
                                                  colliders = list()
<u>Vec3</u>
                                                  auto_render = False
                                                  outline = Entity(parent=self, model=Quad(segments=0, mode='line',
Animation
                                                  thickness=1), color=color.cyan, z=.01, origin=(-.5,-.5),
Animator
                                                  enabled=self.edit mode)
<u>ButtonGroup</u>
                                                  uv_dict = {
ButtonList
                                                           '11111111' : [(4,1), (5,1), (6,1), (7,1)],
                                                                                                                # fill
<u>Node</u>
                                                  single\_block\_coordinates = [(4,0), (5,0), (6,0), (7,0)]
Conversation
                                                  variation_chance = [0,0,0,0,1,1,1,2,2,3]
Cursor
                                                  uv_margin = .002
DebugMenu
<u>Draggable</u>
                                                  update()
<u>DropdownMenuButton</u>
                                                  draw_temp(position)
<u>DropdownMenu</u>
                                                  input(key)
<u>EditorCamera</u>
                                                  render()
<u>ExitButton</u>
                                                  save()
FileButton
<u>FileBrowser</u>
                                                    EditorCamera()
<u>FileBrowserSave</u>
                                                    tilemap = Tilemap('tilemap_test_level', tileset='test_tileset',
<u>FirstPersonController</u>
                                                    tileset_size=(8,4), parent=scene)
FrameAnimation3d
                                                    camera.orthographic = True
<u>GridEditor</u>
                                                    camera.position = tilemap.tilemap.size / 2
<u>PixelEditor</u>
                                                    camera.fov = tilemap.tilemap.height
ASCIIEditor
```

<u>HealthBar</u> <u>HotReloader</u> 'move\_line\_up':

('ctrl+up arrow', 'ctrl+up arrow hold'),

```
Text('press tab to toggle edit mode', origin=(.5,0), position=
                                               (-.55,.4))
light dark
                                            Tooltip(Text)
                                              ursina.prefabs.tooltip
                                              Tooltip(text='', background_color=color.black66, **kwargs)
<u>application</u>
                                              original scale = self.scale
                                              update()
                                                app = Ursina()
input handler
                                                    tooltip_test = Tooltip(
                                                    '<scale:1.5><pink>' + 'Rainstorm' + '<scale:1> \n \n' +
mesh_importer
                                                '''Summon a <blue>rain
                                                storm <default>to deal 5 <blue>water
                                                damage <default>to <red>everyone, <default>including
string_utilities
                                                <orange>yourself. <default>
                                                Lasts for 4 rounds.'''.replace('\n', ''),
<u>texture_importer</u>
                                                        background_color=color.red
                                                )
<u>ursinastuff</u>
                                                    tooltip_test.enabled = True
                                                    app.run()
BoxCollider
<u>SphereCollider</u>
<u>MeshCollider</u>
                                            TrailRenderer(Entity)
<u>CollisionZone</u>
                                              ursina.prefabs.trail_renderer
CubicBezier
                                              TrailRenderer(thickness=10, color=color.white, end_color=color.clear,
                                              length=6, **kwargs)
DirectionalLight
                                              renderer = Entity(
                                                      mode1
<u>AmbientLight</u>
                                              update_step = .025
                                              update()
                                              on destroy()
                                                window.color = color.black
                                                mouse.visible = False
                                                player = Entity()
                                                player.graphics = Entity(parent=player, scale=.1, model='circle')
                                                trail_renderer = TrailRenderer(parent=player, thickness=100,
                                                color=color.yellow, length=6)
<u>LoopingList</u>
                                                pivot = Entity(parent=player)
                                                trail renderer = TrailRenderer(parent=pivot, x=.1, thickness=20,
                                                color=color.orange)
                                                trail_renderer = TrailRenderer(parent=pivot, y=1, thickness=20,
                                                color=color.orange)
                                                trail_renderer = TrailRenderer(parent=pivot, thickness=2,
ButtonGroup
                                                color=color.orange, alpha=.5, position=(.4,.8))
                                                trail_renderer = TrailRenderer(parent=pivot, thickness=2,
                                                color=color.orange, alpha=.5, position=(-.5,.7))
Conversation
                                                def update():
                                                    player.position = lerp(player.position, mouse.position*10,
                                                time.dt*4)
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
                                                    if pivot:
<u>EditorCamera</u>
                                                        pivot.rotation_z -= 3
                                                        pivot.rotation_x -= 2
<u>FileBrowser</u>
                                                def input(key):
<u>FileBrowserSave</u>
                                                    if key == 'space':
<u>FirstPersonController</u>
```

destroy(pivot)

**Entity** <u>Text</u>

**Button** 

mouse <u>raycaster</u>

<u>build</u> camera

color curve

<u>main</u>

<u>scene</u>

text

shader

<u>window</u>

<u>Audio</u> Collider

Color

<u>Keys</u>

<u>Light</u>

HitInfo

**PointLight** 

<u>SpotLight</u> <u>Ursina</u>

<u>MeshModes</u>

<u>Sequence</u>

<u>Shader</u>

<u>Texture</u>

<u>Trigger</u>

<u>Empty</u>

Vec2

<u>Vec3</u>

Node

Cursor

**DebugMenu** 

<u>Draggable</u>

**ExitButton** 

FileButton

FrameAnimation3d <u>GridEditor</u> <u>PixelEditor</u> **ASCIIEditor** <u>HealthBar</u> <u>HotReloader</u>

**Animation** 

**ButtonList** 

**Animator** 

<u>Mesh</u> <u>Wait</u>

<u>Func</u>

<u>ursinamath</u>

<u>duplicate</u>

```
<u>light</u> <u>dark</u>
Entity
<u>Text</u>
Button
mouse
<u>raycaster</u>
<u>application</u>
build
camera
color
curve
<u>duplicate</u>
input handler
<u>main</u>
mesh_importer
<u>scene</u>
shader
string_utilities
text
<u>texture_importer</u>
<u>ursinamath</u>
ursinastuff
<u>window</u>
<u>Audio</u>
Collider
<u>BoxCollider</u>
<u>SphereCollider</u>
<u>MeshCollider</u>
<u>CollisionZone</u>
Color
CubicBezier
HitInfo
<u>Keys</u>
<u>Light</u>
DirectionalLight
PointLight
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
<u>Mesh</u>
<u>Wait</u>
<u>Func</u>
<u>Sequence</u>
<u>Shader</u>
<u>Texture</u>
<u>Trigger</u>
<u>Empty</u>
LoopingList
Vec2
<u>Vec3</u>
Animation
Animator
<u>ButtonGroup</u>
ButtonList
Node
Conversation
Cursor
DebugMenu
<u>Draggable</u>
<u>DropdownMenuButton</u>
DropdownMenu
<u>EditorCamera</u>
<u>ExitButton</u>
FileButton
<u>FileBrowser</u>
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
FrameAnimation3d
<u>GridEditor</u>
<u>PixelEditor</u>
ASCIIEditor
<u>HealthBar</u>
```

## ${\sf SynthGUI}(\underline{\sf Entity})$

<u>ursina.prefabs.ursfx</u>

app.run()

```
SynthGUI(**kwargs)
```

```
wave_panel = Entity(parent=self, scale=.35, x=-0)
waveform = Entity(parent=self.wave_panel, scale_y=.75)
waveform_bg = Entity(parent=self.waveform, model='quad', origin=
(-.5,-.5), z=.01, color=color.black66)
volume_slider = Slider(parent=self.wave_panel, x=-.05, vertical=True,
scale=1.95, min=.05, max=1, default=.75, step=.01,
on value changed=self.play)
wave_selector = ButtonGroup(('sine', 'triangle', 'square', 'noise'),
parent=self.wave_panel, scale=.11, y=-.075)
pitch slider = Slider(parent=self.wave panel, y=-.25, scale=1.95,
min=-36, max=36, default=0, step=1, on_value_changed=self.play,
text='pitch')
pitch_change_slider = Slider(parent=self.wave_panel, y=-.325,
scale=1.95, min=-12, max=12, default=0, step=1,
on_value_changed=self.play, text='pitch change')
speed_slider = Slider(parent=self.wave_panel, scale=1.95, min=.5,
max=4, default=1, step=.1, on_value_changed=self.play, y=-.4,
text='speed')
coin_button = Button(text='coin', parent=self.wave_panel, scale=(.25,
.125), origin=(-.5,.5), y=-.45, on_click=coin_sound)
knobs = [Draggable(parent=self.waveform, scale=.05, model='circle',
color=color.light_gray, highlight_color=color.azure
position=default_positions[i], i=i, min_y=0, max_y=1) for i in
line = Entity(parent=self.waveform, model=Mesh(vertices=[Vec3(0,0,0),
Vec3(1,0,0)], mode='line', thickness=3), z=.01,
color=color.light gray)
bg = Entity(parent=self.wave_panel, model='wireframe_quad', origin=
(-.5,-.5), z=.02, color=color.black, scale_x=1)
play_button = Button(text='>', parent=self.wave_panel, model='circle',
scale=(.125, .125), color=color.azure, origin=(-.5,-.5), position=
(-.075,1.025), on_click=self.play)
copy_button = Button(text='copy', parent=self.wave_panel, scale=(.25,
.125), color=color.dark_gray, origin=(-.5,-.5), position=
(-.075+.125+.025, 1.025), on_click=self.copy_code)
paste_button = Button(text='paste', parent=self.wave_panel, scale=
(.25, .125), color=color.dark_gray, origin=(-.5,-.5), position=
(self.copy_button.x + self.copy_button.scale_x + .025, 1.025),
on_click=self.paste_code)
code_text = Text('', parent=self.wave_panel, scale=2, position=
(self.paste button.x+self.paste button.scale x+.025,1.11))
background_panel = Entity(model=Quad(radius=.025),
parent=self.wave_panel, color=color.black66, z=1, origin=(-.5,-.5),
scale=(1.125,1.75+.025), position=(-.1,-.6))
recipe
coin sound()
drag(this knob=knob)
drop(this knob=knob)
update()
input(key)
copy_code()
paste_code(code="")
draw()
play()
  app = Ursina()
  gui = SynthGUI(enabled=False)
 def toggle_gui_input(key):
    if key == 'f3':
          gui.enabled = not gui.enabled
  Entity(input=toggle_gui_input)
     __name__ == '__main__':
Sprite('shore', z=10, ppu=64, color=color.gray)
gui.enabled = True
```

```
VideoRecorder(Entity)
                                                 ursina.prefabs.video_recorder
light dark
                                                 VideoRecorder(duration=5, name='untitled_video', **kwargs)
Entity
                                                 recording = False
<u>Text</u>
                                                 file_path = Path(application.asset_folder) / 'video temp'
Button
mouse
                                                 duration = duration
<u>raycaster</u>
                                                 fps = 30
                                                 video_name = name
<u>application</u>
build
                                                 max frames = int(self.duration * self.fps)
camera
                                                 frames = []
color
curve
                                                 start_recording()
<u>duplicate</u>
                                                 stop_recording()
input handler
                                                 update()
<u>main</u>
                                                 convert_to_gif()
mesh_importer
<u>scene</u>
                                                   window.size *= .5
shader
                                                   from ursina.prefabs.first_person_controller import
string_utilities
                                                   FirstPersonController
text
                                                   from ursina.shaders import lit_with_shadows_shader
<u>texture_importer</u>
                                                   random.seed(∅)
<u>ursinamath</u>
                                                   Entity.default_shader = lit_with_shadows_shader
<u>ursinastuff</u>
<u>window</u>
                                                   ground = Entity(model='plane', collider='box', scale=64,
                                                   texture='grass', texture_scale=(4,4))
<u>Audio</u>
Collider
                                                   editor_camera = EditorCamera(enabled=False, ignore_paused=True)
<u>BoxCollider</u>
                                                   player = FirstPersonController(model='cube', z=-10,
<u>SphereCollider</u>
                                                   color=color.orange, origin_y=-.5, speed=8)
<u>MeshCollider</u>
                                                   player.collider = BoxCollider(player, Vec3(0,1,0), Vec3(1,2,1))
<u>CollisionZone</u>
Color
                                                   gun = Entity(model='cube', parent=camera, position=(.5,-.25,.25),
CubicBezier
                                                   scale=(.3,.2,1), origin_z=-.5, color=color.red, on_cooldown=False)
HitInfo
<u>Keys</u>
                                                   shootables_parent = Entity()
<u>Light</u>
                                                   mouse.traverse_target = shootables_parent
DirectionalLight
PointLight
                                                   for i in range(16):
<u>AmbientLight</u>
                                                        Entity(model='cube', origin_y=-.5, scale=2, texture='brick',
<u>SpotLight</u>
                                                   texture_scale=(1,2),
<u>Ursina</u>
                                                            x=random.uniform(-8,8),
<u>MeshModes</u>
                                                            z=random.uniform(-8,8) + 8,
<u>Mesh</u>
<u>Wait</u>
                                                            collider='box',
                                                            scale_y = random.uniform(2,3),
<u>Func</u>
                                                            color=color.hsv(0, 0, random.uniform(.9, 1))
<u>Sequence</u>
Shader
<u>Texture</u>
<u>Trigger</u>
                                                   sun = DirectionalLight()
<u>Empty</u>
                                                   sun.look_at(Vec3(1,-1,-1))
LoopingList
                                                   Sky()
Vec2
<u>Vec3</u>
                                                   vr = VideoRecorder(duration=2)
                                                   def input(key):
Animation
                                                       if key == '5':
Animator
                                                            vr.start_recording()
<u>ButtonGroup</u>
                                                       if key == '6':
ButtonList
                                                            vr.stop_recording()
Node
Conversation
Cursor
DebugMenu
<u>Draggable</u>
<u>DropdownMenuButton</u>
                                               VideoRecorderUI(WindowPanel)
<u>DropdownMenu</u>
                                                 ursina.prefabs.video recorder
<u>EditorCamera</u>
<u>ExitButton</u>
                                                 VideoRecorderUI(**kwargs)
FileButton
<u>FileBrowser</u>
<u>FileBrowserSave</u>
```

<u>FirstPersonController</u>

FrameAnimation3d

<u>GridEditor</u>

<u>PixelEditor</u>

**ASCIIEditor** 

<u>HotReloader</u>

<u>HealthBar</u>

```
duration_label = Text('duration:')
duration_field = InputField(default_value='5')
fps_label = Text('fps:')
fps_field = InputField(default_value='30')
name_label = Text('name:')
name_field = InputField(default_value='untitled_video')
start_button = Button(text='Start Recording [Shift+F12]',
color=color.azure, on_click=self.start_recording)
```

```
visible = False
light dark
                                                  input(key)
                                                  start_recording()
Entity
                                                    window.size *= .5
<u>Text</u>
Button
                                                     from ursina.prefabs.first_person_controller import
                                                    FirstPersonController
mouse
<u>raycaster</u>
                                                    from ursina.shaders import lit_with_shadows_shader
                                                    random.seed(0)
<u>application</u>
                                                    Entity.default_shader = lit_with_shadows_shader
build
camera
                                                    ground = Entity(model='plane', collider='box', scale=64,
color
                                                    texture='grass', texture_scale=(4,4))
curve
                                                    editor camera = EditorCamera(enabled=False, ignore_paused=True)
<u>duplicate</u>
                                                    player = FirstPersonController(model='cube', z=-10,
input handler
                                                    color=color.orange, origin_y=-.5, speed=8)
<u>main</u>
                                                    player.collider = BoxCollider(player, Vec3(0,1,0), Vec3(1,2,1))
mesh_importer
<u>scene</u>
                                                     gun = Entity(model='cube', parent=camera, position=(.5,-.25,.25),
<u>shader</u>
                                                    scale=(.3,.2,1), origin_z=-.5, color=color.red, on_cooldown=False)
string_utilities
text
<u>texture_importer</u>
                                                     shootables_parent = Entity()
<u>ursinamath</u>
                                                    mouse.traverse_target = shootables_parent
<u>ursinastuff</u>
<u>window</u>
                                                     for i in range(16):
                                                         Entity(model='cube', origin_y=-.5, scale=2, texture='brick',
<u>Audio</u>
                                                    texture_scale=(1,2),
                                                              x=random.uniform(-8,8),
Collider
<u>BoxCollider</u>
                                                              z=random.uniform(-8,8) + 8,
                                                              collider='box',
<u>SphereCollider</u>
<u>MeshCollider</u>
                                                              scale_y = random.uniform(2,3),
<u>CollisionZone</u>
                                                              color=color.hsv(0, 0, random.uniform(.9, 1))
Color
CubicBezier
HitInfo
                                                     sun = DirectionalLight()
<u>Keys</u>
<u>Light</u>
                                                    sun.look_at(Vec3(1,-1,-1))
DirectionalLight
                                                    Sky()
PointLight
                                                    vr = VideoRecorder(duration=2)
<u>AmbientLight</u>
                                                    def input(key):
<u>SpotLight</u>
                                                         if key == '5':
<u>Ursina</u>
<u>MeshModes</u>
                                                              vr.start_recording()
<u>Mesh</u>
                                                         if key == '6':
<u>Wait</u>
                                                              vr.stop_recording()
<u>Func</u>
<u>Sequence</u>
Shader
<u>Texture</u>
<u>Trigger</u>
                                                Space()
<u>Empty</u>
                                                  ursina.prefabs.window_panel
LoopingList
Vec2
                                                  Space(height=1)
<u>Vec3</u>
                                                  height = height
Animation
Animator
<u>ButtonGroup</u>
ButtonList
                                                    WindowPanel is an easy way to create UI. It will automatically layout
Node
                                                    the content.
Conversation
Cursor
                                                    wp = WindowPanel(
DebugMenu
                                                         title='Custom Window',
<u>Draggable</u>
                                                         content=(
<u>DropdownMenuButton</u>
                                                              Text('Name:'),
DropdownMenu
                                                              InputField(name='name_field'),
<u>EditorCamera</u>
                                                              Button(text='Submit', color=color.azure),
<u>ExitButton</u>
                                                              Slider(),
FileButton
                                                              Slider(),
<u>FileBrowser</u>
<u>FileBrowserSave</u>
                                                              popup=True,
<u>FirstPersonController</u>
                                                              enabled=False
FrameAnimation3d
<u>GridEditor</u>
<u>PixelEditor</u>
                                                    def input(key):
ASCIIEditor
                                                         if key == 'space':
<u>HealthBar</u>
                                                              wp.enabled = True
<u>HotReloader</u>
```

```
light dark
                                              WindowPanel(Draggable)
Entity
                                                 ursina.prefabs.window panel
<u>Text</u>
Button
                                                 WindowPanel(title='', content=[], **kwargs)
mouse
<u>raycaster</u>
                                                 content = content
                                                 text = title
<u>application</u>
                                                 popup = False
<u>build</u>
                                                 panel = Entity(parent=self, model='quad', origin=(0,.5), z=.1,
camera
                                                 color=self.color.tint(.1), collider='box')
color
curve
                                                 layout()
<u>duplicate</u>
                                                 on_enable()
input handler
                                                 close()
mesh_importer
<u>scene</u>
                                                   WindowPanel is an easy way to create UI. It will automatically layout
shader
                                                   the content.
string_utilities
text
                                                   wp = WindowPanel(
<u>texture_importer</u>
                                                       title='Custom Window',
<u>ursinamath</u>
                                                       content=(
ursinastuff
                                                            Text('Name:'),
<u>window</u>
                                                            InputField(name='name_field'),
                                                            Button(text='Submit', color=color.azure),
<u>Audio</u>
                                                            Slider(),
Collider
                                                            Slider(),
BoxCollider
<u>SphereCollider</u>
                                                            popup=True,
<u>MeshCollider</u>
                                                            enabled=False
<u>CollisionZone</u>
Color
CubicBezier
                                                   def input(key):
HitInfo
                                                       if key == 'space':
<u>Keys</u>
                                                            wp.enabled = True
<u>Light</u>
DirectionalLight
PointLight
<u>AmbientLight</u>
SpotLight
                                               chunk mesh
<u>Ursina</u>
<u>MeshModes</u>
                                                 ursina.scripts.chunk mesh
<u>Mesh</u>
Wait
                                                 app = Ursina()
<u>Func</u>
                                                 t = time.time()
<u>Sequence</u>
                                                 application.asset_folder = application.asset_folder.parent.parent
<u>Shader</u>
                                                 terrain = Entity(model=Terrain('grass_fields_heightmap', skip=8),
<u>Texture</u>
                                                 texture='grass', texture_scale=(3,3), scale=256)
<u>Trigger</u>
                                                 grid = [[None for z in range(8)] for x in range(8)] # make 2d array of
<u>Empty</u>
                                                 entities
LoopingList
                                                 x slices = 8
Vec2
Vec3
                                                 terrain.model.generated_vertices = [v+Vec3(.5,0.5) for v in
                                                 terrain.model.generated_vertices]
Animation
                                                 player = FirstPersonController(position=(0,200,0))
Animator
                                                 player.add_script(NoclipMode())
ButtonGroup
                                                 app.run()
ButtonList
<u>Node</u>
                                                 update()
Conversation
Cursor
DebugMenu
Draggable
<u>DropdownMenuButton</u>
                                               colorize
<u>DropdownMenu</u>
                                                 ursina.scripts.colorize
EditorCamera
ExitButton
FileButton
                                                 get_world_normals(model)
<u>FileBrowser</u>
                                                 colorize(model, left=color.white, right=color.blue, down=color.red,
<u>FileBrowserSave</u>
                                                 up=color.green, back=color.white, forward=color.white, smooth=True,
<u>FirstPersonController</u>
                                                 world_space=True, strength=1)
FrameAnimation3d
<u>GridEditor</u>
                                                   import random
PixelEditor
ASCIIEditor
                                                   for i in range(10):
                                                       e = Entity(model=load_model('sphere',
HealthBar
```

```
path=application.internal_models_compressed_folder,
                                                   use_deepcopy=True))
                                                        e.position =
                                                    (random.uniform(-3,3),random.uniform(-3,3),random.uniform(-3,3))
light dark
                                                        e.rotation
                                                    (random.uniform(0,360),random.uniform(0,360),random.uniform(0,360))
Entity
<u>Text</u>
                                                        e.scale = random.uniform(1,3)
Button
                                                        e.model.colorize(smooth=False, world_space=True, strength=.5)
mouse
<u>raycaster</u>
                                                   Sky(color=color.gray)
<u>application</u>
                                                   EditorCamera()
build
camera
color
curve
<u>duplicate</u>
                                               combine
input handler
                                                 ursina.scripts.combine
<u>main</u>
mesh_importer
                                                 temp_entity = None
<u>scene</u>
<u>shader</u>
                                                 combine(entity, analyze=False, auto_destroy=True, ignore=[])
string_utilities
                                                 get vertices(entity, relative to=None)
text
<u>texture_importer</u>
                                                   p = Entity()
<u>ursinamath</u>
                                                   e1 = Entity(parent=p, model='sphere', y=1.5, color=color.pink)
ursinastuff
                                                   e2 = Entity(parent=p, model='cube', color=color.yellow, x=1,
<u>window</u>
                                                   origin_y=-.5)
                                                   e3 = Entity(parent=e2, model='cube', color=color.yellow, y=2,
<u>Audio</u>
                                                   scale=.5)
Collider
BoxCollider
                                                   def input(key):
<u>SphereCollider</u>
                                                        if key == 'space':
<u>MeshCollider</u>
                                                            from time import perf_counter
<u>CollisionZone</u>
                                                            t = perf_counter()
Color
                                                            p.combine()
CubicBezier
                                                            print('combined in:', perf_counter() - t)
HitInfo
<u>Keys</u>
<u>Light</u>
<u>DirectionalLight</u>
                                                   EditorCamera()
PointLight
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
                                               generate normals
<u>Mesh</u>
<u>Wait</u>
                                                 ursina.scripts.generate_normals
<u>Func</u>
<u>Sequence</u>
Shader
                                                 normalize_v3(arr)
<u>Texture</u>
                                                 generate_normals(vertices, triangles=None, smooth=True)
<u>Trigger</u>
<u>Empty</u>
LoopingList
                                                        (-0.0, -0.5, 0.0), (0.1, -0.48, -0.073), (-0.038, -0.48, -0.11),
Vec2
                                                        (0.361804, -0.22, -0.26), (0.3, -0.32, -0.22), (0.40, -0.25,
<u>Vec3</u>
                                                    -0.14),
                                                        (-0.0, -0.5, 0.0), (-0.038, -0.48, -0.11), (-0.03, -0.48, -0.11)
Animation
Animator
                                                   norms = generate_normals(vertices)
<u>ButtonGroup</u>
ButtonList
Node
Conversation
Cursor
                                               grid layout
DebugMenu
                                                 ursina.scripts.grid_layout
<u>Draggable</u>
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
                                                 grid_layout(1, max_x=8, max_y=8, spacing=(0,0,0), origin=(-.5,.5,0),
<u>EditorCamera</u>
                                                 offset=(0,0,0)
<u>ExitButton</u>
FileButton
                                                   center = Entity(model='quad', scale=.1, color=color.red)
<u>FileBrowser</u>
                                                   p = Entity()
<u>FileBrowserSave</u>
                                                   for i in range(4*5):
<u>FirstPersonController</u>
                                                        b = Button(parent=p, model='quad', scale=.5, scale_x=1,
FrameAnimation3d
                                                   text=str(i), color=color.tint(color.random_color(),-.6))
<u>GridEditor</u>
                                                        b.text_entity.world_scale = 1
<u>PixelEditor</u>
ASCIIEditor
                                                   t = time.time()
                                                   grid_layout(p.children, max_x=7, max_y=10, origin=(0, .5))
HealthBar
<u>HotReloader</u>
```

```
center = Entity(parent=camera.ui, model=Circle(), scale=.005,
                                                    color=color.lime)
                                                    EditorCamera()
light dark
                                                   print(time.time() - t)
Entity
<u>Text</u>
Button
                                                merge_vertices
mouse
<u>raycaster</u>
                                                  ursina.scripts.merge_vertices
<u>application</u>
build
                                                  distance(a, b)
camera
                                                  merge_overlapping_vertices(vertices, triangles=None, max_distance=.1)
color
curve
                                                    verts = ((0,0,0), (1,0,0), (1,1,0), (0,0,0), (1,1,0), (0,1,0))
<u>duplicate</u>
                                                    tris = (0,1,2,3,4,5)
input handler
<u>main</u>
                                                    new_verts, new_tris = merge_overlapping_vertices(verts, tris)
mesh_importer
                                                    print('verts:', (verts), (new_verts))
<u>scene</u>
                                                    print('tris:', (tris), (new_tris))
<u>shader</u>
string_utilities
text
                                                    e = Entity(model=Mesh(new_verts, new_tris, mode='triangle'))
<u>texture_importer</u>
                                                    EditorCamera()
<u>ursinamath</u>
<u>ursinastuff</u>
<u>window</u>
<u>Audio</u>
                                                NoclipMode
Collider
<u>BoxCollider</u>
                                                  ursina.scripts.noclip_mode
<u>SphereCollider</u>
<u>MeshCollider</u>
                                                  NoclipMode(speed=10, require_key='shift')
<u>CollisionZone</u>
Color
                                                  speed = speed
CubicBezier
                                                  require_key = require_key
<u>HitInfo</u>
                                                  ignore_paused = True
<u>Keys</u>
<u>Light</u>
                                                  input(key)
DirectionalLight
                                                  update()
PointLight
<u>AmbientLight</u>
                                                    player = Entity(model='cube', color=color.orange)
<u>SpotLight</u>
                                                    Entity(model='plane', scale=10)
<u>Ursina</u>
                                                    EditorCamera()
<u>MeshModes</u>
<u>Mesh</u>
<u>Wait</u>
                                                    player.add_script(NoclipMode2d())
<u>Func</u>
<u>Sequence</u>
Shader
<u>Texture</u>
<u>Trigger</u>
                                                NoclipMode2d
<u>Empty</u>
                                                  ursina.scripts.noclip_mode
LoopingList
Vec2
                                                  NoclipMode2d(speed=10, require_key='shift')
<u>Vec3</u>
                                                  speed = speed
Animation
                                                  require_key = require_key
Animator
                                                  ignore_paused = True
ButtonGroup
ButtonList
                                                  input(key)
<u>Node</u>
                                                  update()
Conversation
Cursor
                                                    player = Entity(model='cube', color=color.orange)
DebugMenu
                                                    Entity(model='plane', scale=10)
<u>Draggable</u>
                                                    EditorCamera()
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
<u>EditorCamera</u>
                                                    player.add script(NoclipMode2d())
<u>ExitButton</u>
FileButton
<u>FileBrowser</u>
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
FrameAnimation3d
                                                PositionLimiter()
<u>GridEditor</u>
                                                  ursina.scripts.position_limiter
<u>PixelEditor</u>
ASCIIEditor
                                                  PositionLimiter(min_x=-math.inf, max_x=math.inf, min_y=-math.inf,
<u>HealthBar</u>
                                                  max_y=math.inf, min_z=-math.inf, max_z=math.inf)
```

```
min_x = min_x
                                                   max_x = max_x
light dark
                                                   min y = min y
                                                   max_y = max_y
                                                   min_z = min_z
Entity
<u>Text</u>
                                                   max_z = max_z
Button
mouse
                                                   update()
<u>raycaster</u>
<u>application</u>
build
camera
                                                 project uvs
color
                                                   ursina.scripts.project_uvs
curve
<u>duplicate</u>
input handler
                                                   project_uvs(model, aspect_ratio=1, direction='forward',
<u>main</u>
                                                   regenerate=False)
mesh_importer
<u>scene</u>
                                                     e = Entity(model='sphere', texture='ursina_logo')
<u>shader</u>
                                                     project uvs(e.model)
string_utilities
                                                     EditorCamera()
text
<u>texture_importer</u>
<u>ursinamath</u>
<u>ursinastuff</u>
<u>window</u>
                                                 Scrollable()
<u>Audio</u>
                                                   ursina.scripts.scrollable
Collider
<u>BoxCollider</u>
                                                   Scrollable(**kwargs)
<u>SphereCollider</u>
<u>MeshCollider</u>
                                                   max = inf
<u>CollisionZone</u>
                                                   min = -inf
Color
                                                   scroll_speed = .05
CubicBezier
                                                   scroll_smoothing = 16
HitInfo
                                                   axis = 'y'
<u>Keys</u>
                                                   target value = None
<u>Light</u>
DirectionalLight
                                                   update()
PointLight
                                                   input(key)
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
                                                     This will make target entity move up or down when you hover the
<u>MeshModes</u>
                                                     entity/its children
<u>Mesh</u>
                                                     while scrolling the scroll wheel.
<u>Wait</u>
<u>Func</u>
<u>Sequence</u>
                                                     p = Button(model='quad', scale=(.4, .8), collider='box')
<u>Shader</u>
                                                     for i in range(8):
<u>Texture</u>
                                                          Button(parent=p , scale_y=.05, text=f'giopwjoigjwr{i}',
<u>Trigger</u>
                                                     origin_y=.5, y=.5-(i*.05))
<u>Empty</u>
LoopingList
                                                     p.add_script(Scrollable())
Vec2
<u>Vec3</u>
Animation
Animator
                                                 SmoothFollow()
ButtonGroup
                                                   ursina.scripts.smooth_follow
ButtonList
<u>Node</u>
                                                   SmoothFollow(target=None, offset=(0,0,0), speed=8, rotation_speed=0,
Conversation
                                                   rotation_offset=(0,0,0))
Cursor
DebugMenu
                                                   target = target
Draggable
                                                   offset = offset
<u>DropdownMenuButton</u>
                                                   speed = speed
<u>DropdownMenu</u>
                                                   rotation_speed = rotation_speed
<u>EditorCamera</u>
                                                   rotation_offset = rotation_offset
ExitButton
FileButton
                                                   update()
<u>FileBrowser</u>
<u>FileBrowserSave</u>
                                                     player = Entity(model='cube', color=color.orange)
<u>FirstPersonController</u>
FrameAnimation3d
                                                     def update():
<u>GridEditor</u>
                                                          player.x += held_keys['d'] * .1
<u>PixelEditor</u>
                                                          player.x -= held_keys['a'] * .1
ASCIIEditor
<u>HealthBar</u>
<u>HotReloader</u>
```

```
light dark
Entity
<u>Text</u>
Button
mouse
<u>raycaster</u>
<u>application</u>
<u>build</u>
camera
color
curve
<u>duplicate</u>
input handler
<u>main</u>
mesh_importer
<u>scene</u>
shader
string_utilities
<u>text</u>
<u>texture_importer</u>
<u>ursinamath</u>
<u>ursinastuff</u>
<u>window</u>
<u>Audio</u>
Collider
<u>BoxCollider</u>
<u>SphereCollider</u>
<u>MeshCollider</u>
<u>CollisionZone</u>
Color
<u>CubicBezier</u>
HitInfo
<u>Keys</u>
<u>Light</u>
DirectionalLight
PointLight
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
<u>Mesh</u>
<u>Wait</u>
<u>Func</u>
<u>Sequence</u>
<u>Shader</u>
<u>Texture</u>
<u>Trigger</u>
<u>Empty</u>
LoopingList
Vec2
<u>Vec3</u>
Animation
Animator
ButtonGroup
ButtonList
<u>Node</u>
Conversation
Cursor
DebugMenu
<u>Draggable</u>
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
EditorCamera
ExitButton
FileButton
<u>FileBrowser</u>
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
FrameAnimation3d
<u>GridEditor</u>
<u>PixelEditor</u>
ASCIIEditor
<u>HealthBar</u>
<u>HotReloader</u>
```

```
e = Entity(model='cube')
sf = e.add_script(SmoothFollow(target=player, offset=(0,2,0)))

def input(key):
    global sf
    if key == '1' and sf in e.scripts:
        e.scripts.remove(sf)

EditorCamera()
```

#### terraincast

#### models

```
'arrow'
'circle'
'cube'
'cube_uv_top'
'diamond'
'icosphere'
'line'
'plane'
'quad'
'scale_gizmo'
'sky_dome'
'sphere'
'wireframe_cube'
'wireframe_quad'
```

### textures

```
'arrow_down'
'arrow_right'
'brick'
'circle'
'circle_outlined'
'cog'
'cursor'
'file_icon'
'folder'
'grass'
'heightmap_1'
'horizontal_gradient'
```

```
'noise
                                                    'radial_gradient'
                                                   'rainbow'
light dark
                                                   'reflection map 3'
                                                   'shore'
Entity
                                                   'sky_default'
<u>Text</u>
                                                    'sky_sunset'
Button
                                                    'test_tileset'
                                                    'tilemap_test_level'
mouse
                                                   'ursina_logo'
<u>raycaster</u>
                                                   'ursina_wink_0000'
<u>application</u>
                                                   'ursina_wink_0001'
                                                   'vertical_gradient'
build
camera
                                                    'vignette'
color
                                                    'white cube'
curve
<u>duplicate</u>
input handler
                                                   e = Entity(model='cube', texture='brick')
<u>main</u>
mesh_importer
<u>scene</u>
<u>shader</u>
string_utilities
                                                 shaders
text
                                                   basic_lighting_shader
<u>texture_importer</u>
                                                   colored_lights_shader
<u>ursinamath</u>
                                                   fresnel shader
<u>ursinastuff</u>
                                                   geom_shader
<u>window</u>
                                                   lit_with_shadows_shader
                                                   matcap_shader
<u>Audio</u>
                                                   normals shader
Collider
                                                   projector shader
<u>BoxCollider</u>
                                                   texture_blend_shader
<u>SphereCollider</u>
                                                   transition_shader
<u>MeshCollider</u>
                                                   triplanar_shader
<u>CollisionZone</u>
                                                   unlit_shader
Color
                                                   camera_contrast
CubicBezier
                                                   camera_empty
<u>HitInfo</u>
                                                   camera_grayscale
<u>Keys</u>
                                                   camera_outline_shader
<u>Light</u>
                                                   camera_vertical_blur
DirectionalLight
                                                   fxaa
PointLight
                                                   pixelation_shader
<u>AmbientLight</u>
                                                   ssao
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
                                                     from ursina.shaders import normals shader
<u>Mesh</u>
                                                     e = Entity(shader=normals shader)
<u>Wait</u>
<u>Func</u>
<u>Sequence</u>
Shader
<u>Texture</u>
                                                 Circle(Mesh)
<u>Trigger</u>
<u>Empty</u>
                                                   ursina.models.procedural.circle
LoopingList
Vec2
                                                   Circle(resolution=16, radius=.5, rotate=True, mode='ngon', **kwargs)
Vec3
                                                   vertices = list()
Animation
Animator
<u>ButtonGroup</u>
                                                     e = Entity(model=Circle(8, mode='line', thickness=10),
ButtonList
                                                     color=color(60,1,1,.3))
<u>Node</u>
                                                     print(e.model.recipe)
Conversation
                                                     origin = Entity(model='quad', color=color.orange, scale=(.05, .05))
Cursor
                                                     ed = EditorCamera(rotation_speed = 200, panning_speed=200)
DebugMenu
<u>Draggable</u>
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
EditorCamera
                                                 Cone (Mesh)
<u>ExitButton</u>
                                                   ursina.models.procedural.cone
FileButton
<u>FileBrowser</u>
                                                   Cone(resolution=4, radius=.5, height=1, direction=(0,1,0),
<u>FileBrowserSave</u>
                                                   add_bottom=True, mode='triangle', **kwargs)
<u>FirstPersonController</u>
FrameAnimation3d
<u>GridEditor</u>
<u>PixelEditor</u>
                                                     Entity(model=Cone(8, direction=(0,1,0)),
ASCIIEditor
                                                     color=color(60,1,1,.3))
<u>HealthBar</u>
<u>HotReloader</u>
```

```
origin = Entity(model='quad', color=color.orange, scale=(.05, .05))
                                                    ed = EditorCamera(rotation_speed = 200, panning_speed=200)
light dark
Entity
<u>Text</u>
                                                Cylinder(Pipe)
Button
                                                  ursina.models.procedural.cylinder
mouse
<u>raycaster</u>
                                                  Cylinder(resolution=8, radius=.5, start=0, height=1, direction=
                                                  (0,1,0), mode='triangle', **kwargs)
<u>application</u>
build
camera
color
                                                    Entity(model=Cylinder(6, start=-.5), color=color.color(60,1,1,.3))
curve
                                                    origin = Entity(model='quad', color=color.orange, scale=(5, .05))
<u>duplicate</u>
                                                    ed = EditorCamera(rotation_speed = 200, panning_speed=200)
input handler
<u>main</u>
mesh_importer
<u>scene</u>
<u>shader</u>
                                                Grid(Mesh)
string_utilities
text
                                                  ursina.models.procedural.grid
<u>texture_importer</u>
<u>ursinamath</u>
                                                  Grid(width, height, mode='line', thickness=1, **kwargs)
ursinastuff
<u>window</u>
                                                  width = width
                                                  height = height
<u>Audio</u>
Collider
BoxCollider
                                                  Entity(model=Grid(2, 6))
<u>SphereCollider</u>
<u>MeshCollider</u>
<u>CollisionZone</u>
Color
CubicBezier
                                                Pipe(Mesh)
<u>HitInfo</u>
                                                  ursina.models.procedural.pipe
<u>Keys</u>
<u>Light</u>
                                                  Pipe(base_shape=Quad, origin=(0,0), path=((0,0,0),(0,1,0)),
<u>DirectionalLight</u>
PointLight
                                                  thicknesses=((1,1),), look_at=True, cap_ends=True, mode='triangle',
                                                  **kwargs)
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
                                                  base_shape = base_shape
<u>MeshModes</u>
<u>Mesh</u>
                                                    path = (\text{Vec3}(0,0,0), \text{Vec3}(0,1,0), \text{Vec3}(0,3,0), \text{Vec3}(0,4,0),
<u>Wait</u>
                                                    Vec3(2,5,0))
<u>Func</u>
                                                    thicknesses = ((1,1), (.5,.5), (.75,.75), (.5,.5), (1,1))
<u>Sequence</u>
                                                    e = Entity(model=Pipe(path=path, thicknesses=thicknesses))
Shader
                                                    e.model.colorize()
<u>Texture</u>
<u>Trigger</u>
                                                    EditorCamera()
<u>Empty</u>
                                                    origin = Entity(model='cube', color=color.magenta)
LoopingList
                                                    origin.scale *= .25
Vec2
Vec3
Animation
Animator
<u>ButtonGroup</u>
                                                Plane(Mesh)
ButtonList
                                                  ursina.models.procedural.plane
<u>Node</u>
Conversation
                                                  Plane(subdivisions=(1,1), mode='triangle', **kwargs)
Cursor
DebugMenu
                                                  vertices, self.triangles = list(), list()
<u>Draggable</u>
                                                  uvs = list()
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
<u>EditorCamera</u>
                                                    front = Entity(model=Plane(subdivisions=(3,6)), texture='brick',
<u>ExitButton</u>
                                                    rotation_x=-90)
FileButton
<u>FileBrowser</u>
                                                     _ed = EditorCamera()
<u>FileBrowserSave</u>
                                                    Entity(model='cube', color=color.green, scale=.05)
<u>FirstPersonController</u>
FrameAnimation3d
<u>GridEditor</u>
<u>PixelEditor</u>
```

### QuadMesh (Mesh)

**ASCIIEditor** 

<u>HealthBar</u> <u>HotReloader</u>

```
light dark
Entity
<u>Text</u>
Button
mouse
<u>raycaster</u>
<u>application</u>
build
camera
color
curve
<u>duplicate</u>
input handler
<u>main</u>
mesh_importer
<u>scene</u>
<u>shader</u>
string_utilities
text
<u>texture_importer</u>
<u>ursinamath</u>
ursinastuff
<u>window</u>
<u>Audio</u>
Collider
BoxCollider
<u>SphereCollider</u>
<u>MeshCollider</u>
<u>CollisionZone</u>
Color
CubicBezier
HitInfo
<u>Keys</u>
<u>Light</u>
DirectionalLight
PointLight
<u>AmbientLight</u>
<u>SpotLight</u>
<u>Ursina</u>
<u>MeshModes</u>
<u>Mesh</u>
<u>Wait</u>
<u>Func</u>
<u>Sequence</u>
<u>Shader</u>
<u>Texture</u>
<u>Trigger</u>
<u>Empty</u>
LoopingList
Vec2
<u>Vec3</u>
Animation
Animator
ButtonGroup
ButtonList
<u>Node</u>
Conversation
Cursor
DebugMenu
<u>Draggable</u>
<u>DropdownMenuButton</u>
<u>DropdownMenu</u>
<u>EditorCamera</u>
ExitButton
FileButton
<u>FileBrowser</u>
<u>FileBrowserSave</u>
<u>FirstPersonController</u>
FrameAnimation3d
<u>GridEditor</u>
<u>PixelEditor</u>
ASCIIEditor
<u>HealthBar</u>
<u>HotReloader</u>
```

```
ursina.models.procedural.quad
 QuadMesh(radius=.1, segments=8, aspect=1, scale=(1,1), mode='ngon',
 vertices = [Vec3(0,0,0), Vec3(1,0,0), Vec3(1,1,0), Vec3(0,1,0)]
 radius = radius
 mode = mode
 thickness = thickness
 uvs = list()
 vertices = [(v[0]-offset[0], v[1]-offset[1], v[2]-offset[2]) for v in
 self.vertices]
   from time import perf counter
   t = perf_counter()
   for i in range(100):
       Entity(model=Quad(scale=(3,1), thickness=3, segments=3,
   mode='line'), color = color.color(0,1,1,.7))
   print('-----', (perf_counter() - t))
   origin = Entity(model='quad', color=color.orange, scale=(.05, .05))
   camera.z = -5
Terrain(Mesh)
 ursina.models.procedural.terrain
 Terrain(heightmap, skip=1, **kwargs)
 heightmap = heightmap
 skip = skip
               # should be power of two.
 width, self.depth = self.heightmap.width//skip,
 self.heightmap.height//skip
 aspect_ratio = self.width / self.depth
 height_values = asarray(img)
 height_values = flip(self.height_values, axis=0)
 height_values = swapaxes(self.height_values, 0, 1)
 vertices, self.triangles = list(), list()
 uvs = list()
 normals = list()
 height_values = [[j/255 for j in i] for i in self.height_values]
   e = Entity(model=Terrain('heightmap 1', skip=16), scale=(20,5,20),
   texture='heightmap_1')
   Entity(model='plane', scale=e.scale, color=color.red)
   EditorCamera()
   Sky()
```