ez

The base EZpanda class.
Created in "main.py".
Is added to builtins so is global to all scripts.

```
from scripts.EZpanda.EZ import EZ
ez = EZ(config=config)
```

ez.mask

Used for setting collisions and physics masks. index range 0-32	
`mask = ez.mask[10]`	return bit mask at index 10
`nomask = ez.mask[0] or ez.mask['NONE']`	return no mask
`allmasks = ez.mask['ALL']`	return all masks

ez.PATH

File system path to main.py	**CONST string; (unix style system pathing)**

ez.Node

Base EZpanda node **Inherits from dict**	
`node = ez.Node( panda_node=None, parent=None )`	will create panda_node if None
`name`	string, default: 'EZnode'
`panda_node`	reference to Panda3D node
`hide()`	hide the node
`show()`	show the node
`is_hidden()`	bool
`parent`	ez.Node
`delete()`	remove node and all its childreen
`get_children()`	list
`look_at( node_or_pos )`	face node or point
`get_distance_to( node )`	float
`get_facing_vector()`	vec3
`get_relative_vector( node, vec3 )`	vec3
`apply_transform()`	set current transform as default
`set_shader_input(shader_value_name, value)`	string, object
`set_shader_inputs( **kwargs )`	a=1, b=2, ... or dict {string: object}
`get_rx( node )`	relative x to node
`get_ry( node )`	relative y to node
`get_rz( node )`	relative z to node
`get_rpos( node )`	relative pos to node
`get_rh( node )`	relative h to node
`get_rp( node )`	relative p to node
`get_rr( node )`	relative r to node
`get rhpr( node )`	relative hpr to node
`set_rx( x, node )`	set x relative to node
`set_ry( y, node )`	set y relative to node
`set_rz( x, node )`	set z relative to node
`set_rpos( pos, node )`	set pos relative to node
`set_rh( h, node )`	set h relative to node
`set_rp( p, node )`	set p relative to node

Base EZpanda node **Inherits from dict**	
`set_rr( r, node )`	set r relative to node
`set_rhpr( hpr, node )`	set hpr relative to node
`x`	float
`y`	float
`z`	float
`pos`	Vec3
`h`	float
`p`	float
`r`	float
`hpr`	VBase3
`scale`	float
`copy_render_state( custom_name )`	string; returns a render state
`set_render_state( state )`	render_state
`set_render_state_to_camera( state, camera )`	render_state, camera
`shader`	shader
`depth_write`	bool
`transparency`	ez.flags.transparency.*FLAG*

ez.Model

Inherits from ez.Node	
`model = ez.Model( mesh, parent=None )`	
`get_bounds`	
`get_tight_bounds`	
`show_bounds`	
`show_tight_bounds`	
`hide_bounds`	

ez.Actor

inherits from ez.Model	
`actor = Actor( mesh, animations=none, parent=none )`	
`play( string )`	play animation
`loop( string )`	loop animation
`stop( stirng )`	stop animation
`get_animations`	list

ez.Line

inherits from ez.Node **Used for drawing lines**	
`panda_line`	Panda3D line class
`move_to( vec3 )`	move drawing point
`set_color( color )`	set color of line to draw
`set_thickness( float )`	how thick line should be
`draw_to( vec3 )`	position to draw line to
`reset()`	reset drawing
`create()`	create the line

ez.SoftInstance

Inherits from ez.Node **Used for instancing models**	
`instances = ez.SoftInstance(mesh, total_instances, parent=None)`	
`instances[ index ]`	get model of instance

ez.HardInstance

Inherits from ez.Node **Used for instancing a mesh on the GPU**	
`instances = ez.HardInstance( mesh, total_instances, boundsWHD, HPR=(0,0,0), parent=None)`	
`get_bounds`	
`show_bounds`	
`hide_bounds`	
`get_total_instances`	
`set_instance_pos( index, pos, size=1)`	
`set_instances_pos( index_pos_size )`	list of tuples; [(index, pos, size)]
`generate_random_pos( scale_min=1.0, scale_max=1.0 )`	

ez.Text

Inherits form ez.Node	
text = ez.Text(font, text="", parent=None)	

`A_BOXED_CENTER`
`A_BOXED_LEFT`
`A_BOXED_RIGHT`
`A_CENTER`
`A_LEFT`
`A_RIHT`
	Alignment flags

`RM_DISTANCE_FIELD`
`RM_EXTRUDE`
`RM_INVALID`
`RM_POLYGON`
`RM_SOLID`
`RM_TEXTURE`
`RM_WIREFRAME`
	Render flags

clear_frame()	
clear_card()	
`make_mesh`	Create a mesh from the text
`text`	string
`font`	font
`small_caps`	bool
`small_caps_scale`	float
`slant`	float
`color`	(r, g, b, a)
`shadow`	(x, y)
`shadow_color`	(r, g, b, a)
`wordwrap`	float
`align`	Alignment Flag
`frame_color`	(r, g, b, a)
`frame_width`	int
`frame_corners`	bool
`card_color`	(r, g, b, a)
`card_decal`	bool
`set_card_margin`	(left, right, bottom, top)

ez.Camera

Inherits from ez.Node	
`ORTHO` `PERSPECTIVE`	Lens Flags
`camera = ez.Camera(lens=ez.Camera.PERSPECTIVE, parent=None)`	
`get_projected_ray( aspect2D_pos )`	Returns 3D FROM and TO position from camera lens to camera far: (vec3, vec3)
`add_render_state`(state, state_name)	render_state, str
`create_depth_map()`	returns depth_map
`get_depth_map()`	returns depth_map if one has been created
`fov`	field of view angle
`vfov`	vertical field of view angle
`near`	camera lens position; float
`far`	camera far position; float

ez.TextureBuffer

buffer = ez.TextureBuffer(widht, height, display_region=(0,1,0,1), name="Texture Buffer")	
`camera`	EZ.Camera
`background`	bool
`background_color`	(r,g,b,a)

ez.ProceduralMesh

pmesh = ez.ProceduralMesh(format=ez.ProceduralMesh.V3N3, verts=[], tris=[])	
`set_data( verts, tris )`	list, list
`create_mesh()`	

ez.AudioManager

am = ez.AudioManager(panda_audio=None)	
`load( filename )`	string
`volume`	float
`concurrent_limit`	int
`stop_all_sounds()`	

ez.Audio3DManager

a3Dm = ez.Audio3DManager(audio_manager)	
`load( filename )`	str
`listener`	node
`distance_factor`	float
`drop_off_factor`	float

ez.Vec2

vec = ez.Vec2(x, y)	

ez.Vec3

vec = ez.Vec3(x, y, z)	

ez.Vec4

vec = ez.Vec4(x, y, z, w)	

ez.Point2

point = ez.Point2(x, y)	

ez.Point3

point = ez.Point3(x, y, z)	

ez.Point4

point = ez.Point4(x, y, z, w)	

ez.VBase2

base = ez.VBase2(x, y)	

ez.VBase3

base = ez.VBase3(x, y, z)	

ez.VBase4

base = ez.VBase4(x, y, z, w)	

ez.lights

ez.lgihts.Sun

Inherits from ez.Node	
`sun = ez.lights.Sun( size=(10,10), shadow_size=(512,512), parent=None )`	
`add_render_state( state, state_name)`	render_state, str
`set_shadow_castor( widht, height, dynamic=True)`	
`near`	float
`far`	float

ez.panda_showbase

Panda3D ShowBase	

ez.is_button_down

ez.is_button_down(str)	Returns bool; tests if key is down

ez.run

ez.run()	Run the game

ez.make_task

task = ez.make_task(function, *args, use_task=True, name='EZtask')	

ez.add_task

ez.add_task(task)	

ez.remove_task

ez.remove_taks(task)	

ez.random

class for randomization	
`seed( int )`	
`float()`	returns float between 0.0 - 1.0
`uniform( low, high )`	return float between low - high
`int( low, high )`	return int between low - high
`range( low, high, step)`	
`choice( array )`	return random object from array
`shuffle( list )`	randomize a list

ez.math

class for doing math	
ez.math.distance(vec1, vec2)	float; return distance between two vectors

ez.window

class for accessing system window	
get_size()	
get_aspect_ratio()	
get_aspect2D_edges()	
get_display_mode(int=0)	Get (width, height, rate) of display; 0=first monitor, 1=second monitor, ...
set_display(width, height, rate=60)	
set_max_fps(int)	
fullscreen	bool
show_fps	bool
background_color	(r, g, b, a)

ez.mouse

class for interacting with mouse	
hide()	Hide the mouse cursor
show()	Show the mouse cursor
cursor	cursor
pos	vec2

ez.enable

class for enabling optionals	
particles()	
gamepads()	
collision()	
physics()	

ez.load

class for loading assets	
font(filename)	
sound(filename)	
sound3D(filename)	
gen_sound(filename, instance_count)	generator class of sounds, for playing same sound multiple times
gen_sound3D(filename, instance_count)	
music(filename)	
mesh(filename)	
texture(filename, af=4)	af = anisotropic filter rate
cursor(filename)	
shader(filename)	
scene(name)	

ez.audio

Default audio manager	

ez.audio3D

Default audio 3D manager	

ez.music

Default music manager	

ez.intervals

class for creating intervals	
pos(node, start_pos, end_pos, dureation, blend='noBlend', name=None, relative_to=None, fluid=0, bake_in_start=1)	
hpr(node, start_hpr, end_hpr, duration, blend='noBlend', name=None, relative_to=None, bake_in_start=1)	
Function(func, fr, to, duration, blend='noBlend', args=[], name=None)	

ez.gamepads

class for controlling gamepads, acts as a list holding all gamepads	

ez.particls

class for controlling particles	

ez.collision

class for controlling collisions	

ez.physics

class for controlling physics	

ez.get_dt()

returns delta time, (time since last frame)	

ez.end()

For ending the program	

ez.display_region

Default camera display region	

ez.aspect2D_depth

ez.set_aspect2D_depth(bool)	**Enable depth sorting on aspect2D**

ez.add_input_events

enable eventing for keys	
ez.add_input_events(keys)	list of key names: ['a', 'b', 'esc']

ez.remove_input_events

ez.remove_input_events(keys)	

ez.reset_scene(scene)

Reset scene back to defaults	

ez.set_scene

Set the active scene	
ez.set_sene(scene)	