
pystk Documentation

Release 1.0

Philipp Krähenbühl

Sep 22, 2021

THE BASICS

1	Hardware Requirements	3
2	License	5
2.1	Installation	5
2.2	Quick start	6
2.3	Graphics Setup	7
2.4	Race	8
2.5	Logging	16
	Index	17

This is a heavily modified version of the free [SuperTuxKart](#) racing game for sensorimotor control experiments.

Many parts that make SuperTuxKart fun and entertaining to play were removed, and replaced with a highly efficient and customizable python interface to the game. The python interface supports the full rendering engine and all assets of the original game, but no longer contains any sound playback, networking, or user interface.

If you find a bug in this version of supertuxkart please do not report it to the original project, this project significantly diverged from the original intention of the video game.

HARDWARE REQUIREMENTS

To run SuperTuxKart, make sure that your computer's specifications are equal or higher than the following specifications:

- A graphics card capable of 3D rendering - NVIDIA GeForce 8 series and newer (GeForce 8100 or newer), AMD/ATI Radeon HD 4000 series and newer, Intel HD Graphics 3000 and newer. OpenGL $\geq$ 3.3
- You should have a CPU that's running at 1 GHz or faster.
- You'll need at least 512 MB of free VRAM (video memory).
- Minimum disk space: 800 MB

LICENSE

The software is released under the GNU General Public License (GPL). Information about the licenses for the artwork is contained in `data/licenses`.

2.1 Installation

2.1.1 Using pip

```
pip install PySuperTuxKart
```

2.1.2 From source

```
python setup.py build
python setup.py install
```

2.1.3 Development

Clone the repository <https://github.com/philkr/pystk>. For easier development, it is recommended to install `pystk` directly through `cmake`.

```
mkdir build
cd build
cmake ..
make
```

CMake will place a copy of the library in the top level directly, with allows any examples to run from that directory. Make sure the fetch the game assets, if they don't already exist.

```
python setup.py fetch_data
```

2.1.4 Documentation

```
cd docs
make html
```

PySTK does not compile on readthedocs.org due to some missing dependencies. This means that autodoc does not work there. In order to circumvent this, the autodoc parts of the documentation are split off and can be built using

```
make rst
```

Make sure to build the html again after.

2.2 Quick start

Let's walk through a simple example on how to use pystk. You'll first need to setup the rendering engine. SuperTuxKart uses a lot of global memory objects, some of them should only be initialized once. Hence, you should only setup the rendering engine *once* per process.

```
config = pystk.GraphicsConfig.hd()
config.screen_width = 800
config.screen_height = 600
pystk.init(config)
```

This setup uses the high-definition graphics preset and sets the resolution to 800 x 600.

Now we're ready to start the race. You may play as many races as you want, but you can only run *one* race per process. If you try to start (or setup) a second race before completing the first, the wrapper will raise an exception and eventually crash.

```
config = pystk.RaceConfig()
config.num_kart = 2 # Total number of karts
config.players[0].controller = pystk.PlayerConfig.Controller.AI_CONTROL

config.track = 'lighthouse'

race = pystk.Race(config)
```

This race configuration plays the lighthouse track with a total of 2 karts, one of which is player controlled. By default there is only one player `len(config.players)==1` and all other karts are non-controllable AI karts.

Next, let's start the race and play for a 100 steps.

```
race.start()
for n in range(100):
    race_ended = race.step()
    # Optionally display race.render_data
```

See [Race](#) for a full documentation of the race object and the `render_data`.

Finally, delete the current race object before exiting or starting a new race.

```
race.stop()
del race
```

2.3 Graphics Setup

Before you can use pystk you need to setup the OpenGL rendering engine and graphics settings. There are three default settings `GraphicsConfig::ld` (lowest), `GraphicsConfig::sd` (medium), `GraphicsConfig::hd` (high). Depending on your graphics hardware each setting might perform slightly differently (ld fastest, hd slowest). To setup pystk call:

```
pystk.init(pystk.GraphicsConfig.hd())
# Only call init once per process
... # use pystk
pystk.clean() # Optional, will be called atexit
# Do not call pystk after clean
```

```
class pystk.GraphicsConfig
    SuperTuxKart graphics configuration.

    static hd() → pystk.GraphicsConfig
        High-definitaiton graphics settings

    static ld() → pystk.GraphicsConfig
        Low-definition graphics settings

    static none() → pystk.GraphicsConfig
        Disable graphics and rendering

    static sd() → pystk.GraphicsConfig
        Standard-definition graphics settings

    property animated_characters → bool
        Animate characters

    property bloom → bool
        Enable the bloom effect

    property degraded_IBL → bool
        Disable specular IBL

    property display_adapter → int
        GPU to use (Linux only)

    property dof → bool
        Depth of field effect

    property dynamic_lights → bool
        Enable dynamic lighting

    property glow → bool
        Enable glow around pickup objects

    property high_definition_textures → int
        Enable high definition textures 0 / 2

    property light_shaft → bool
        Enable light shafts

    property mlaa → bool
        Enable anti-aliasing

    property motionblur → bool
        Enable motion blur
```

property particles_effects → int
Particle effect 0 (none) to 2 (full)

property render → bool
Is rendering enabled?

property screen_height → int
Height of the rendering surface

property screen_width → int
Width of the rendering surface

property ssao → bool
Enable screen space ambient occlusion

property texture_compression → bool
Use texture compression

pystk.init(*config*: [pystk.GraphicsConfig](#)) → None
Initialize Python SuperTuxKart. Only call this function once per process. Calling it twice will cause a crash.

pystk.clean() → None
Free Python SuperTuxKart, call this once at exit (optional). Will be called atexit otherwise.

2.4 Race

To start a race create a new Race object. You can configure your race using the RaceConfig object, see [Configuration](#). You need to set the graphics settings before starting a race, see [Graphics Setup](#).

```
pystk.init(pystk.GraphicsConfig.hd())

config = pystk.RaceConfig(track='lighthouse', num_kart=2)
config.players[0].controller = pystk.PlayerConfig.Controller.AI_CONTROL
race = pystk.Race(config)
race.start()

n_steps = 100
for step in range(n_steps):
    race.step() # Use an optional action and set controller to pystk.PlayerConfig.
    ↪ Controller.PLAYER_CONTROL
    # Use race.render_data[0].image
    # Use race.render_data[0].depth
    # Use race.render_data[0].instance
race.stop()
del race
# You may start a new race after you delete the old race object
pystk.clean()
```

class **pystk.Race**
The SuperTuxKart race instance

__init__(*self*: [pystk.Race](#), *config*: [pystk.RaceConfig](#))

restart(*self*: [pystk.Race](#)) → None
Restart the current track. Use this function if the race config does not change, instead of creating a new SuperTuxKart object

start(*self*: [pystk.Race](#)) → None
start the race

step(*args, **kwargs)
Overloaded function.

- **step**(*self*: [pystk.Race](#), *action*: List[[pystk.Action](#)]) → bool
Take a step with an action per agent
- **step**(*self*: [pystk.Race](#), *action*: [pystk.Action](#)) → bool
Take a step with an action for agent 0
- **step**(*self*: [pystk.Race](#)) → bool
Take a step without changing the action

stop(*self*: [pystk.Race](#)) → None
Stop the race

property config → [pystk.RaceConfig](#)
The current race configuration

property render_data → List[[pystk.RenderData](#)]
rendering data from the last step

SuperTuxKart uses several global variables and thus only allows one game instance to run per process. To check if there is already a race running use the `is_running` function.

pystk.is_running() → bool
Is a race running?

2.4.1 Configuration

The player configuration is used to add agents to the race. Each agent can be an AI or player controlled, and produces a separate `render_data` output.

class [pystk.PlayerConfig](#)
SuperTuxKart player configuration

class [Controller](#)

AI_CONTROL = 1

PLAYER_CONTROL = 0

property name → str

property value → int

property controller → [pystk.PlayerConfig.Controller](#)
Let the player (`PLAYER_CONTROL`) or AI (`AI_CONTROL`) drive. The AI ignores actions in `step(action)`.

property kart → str
Kart type, see `list_karts` for a list of kart types

property team → int
Team of the player 0 or 1

The main race configuration specified everything from the track to use, the race type, number of agents and additional AI agents.

```
class pystk.RaceConfig
    SuperTuxKart race configuration.

    class RaceMode

        CAPTURE_THE_FLAG = 5
        FOLLOW_LEADER = 2
        FREE_FOR_ALL = 4
        NORMAL_RACE = 0
        SOCCER = 6
        THREE_STRIKES = 3
        TIME_TRIAL = 1

        property name → str
        property value → int

    property difficulty → int
        Skill of AI players 0..2

    property laps → int
        Number of laps the race runs for

    property mode → pystk.RaceConfig.RaceMode
        Specify the type of race

    property num_kart → int
        Total number of karts, fill the race with num_kart - len(players) AI karts

    property players → pystk.VectorPlayerConfig
        List of all agent players

    property reverse → bool
        Reverse the track

    property seed → int
        Random seed

    property step_size → float
        Game time between different step calls

    property track → str
        Track name

pystk.list_tracks() → List[str]
    Return a list of track names (possible values for RaceConfig.track)

pystk.list_karts() → List[str]
    Return a list of karts to play as (possible values for PlayerConfig.kart)
```

2.4.2 Action

The *Race.step* function takes an optional action or list of actions as an input.

```
class pystk.Action
    SuperTuxKart action

    property acceleration → float
        Acceleration, normalize to 0..1

    property brake → bool
        Hit the brakes. Zero acceleration and brake=True uses reverse gear.

    property drift → bool
        Drift while turning

    property fire → bool
        Fire the current pickup item

    property nitro → bool
        Use nitro

    property rescue → bool
        Call the rescue bird

    property steer → float
        Steering angle, normalize to -1..1
```

2.4.3 Data

```
class pystk.RenderData
    SuperTuxKart rendering output

    property depth → numpy.ndarray
        Depth image of the kart (memoryview[float] screen_height x screen_width)

    property image → numpy.ndarray
        Color image of the kart (memoryview[uint8] screen_height x screen_width x 3)

    property instance → numpy.ndarray
        Instance labels (memoryview[uint32] screen_height x screen_width)
```

Each instance label is spit into an `ObjectType` and instance label. Right shift (`>>`) the instance label by `ObjectType.object_type_shift` to retrieve the object type.

```
class pystk.ObjectType

    object_type_shift
        Number of bits for the instance label (shift of the object type)

    N = 10

    background = 3

    bomb = 6

    kart = 1

    property name → str

    nitro = 5
```

```

object = 7
pickup = 4
projectile = 8
track = 2
unknown = 9
property value → int

```

2.4.4 Game state

PySTK also exposes the internal state of the game.

class `pystk.WorldState`

```

static set_ball_location(position: float3, velocity: float3 = [0.0, 0.0, 0.0], angular_velocity: float3 = [0.0, 0.0, 0.0]) → None

```

Specify the soccer ball / hockey puck position (SOCCER mode only).

```

static set_kart_location(kart_id: int, position: float3, rotation: Quaternion = [0.0, 0.0, 0.0, 1.0], speed: float = 0) → None

```

Move a kart to a specific location.

```

update(self: pystk.WorldState) → None

```

Update this object with the current world state

```

property ffa → pystk.FFA

```

Free for all match info

```

property items → List[pystk.Item]

```

State of items

```

property karts → List[pystk.Kart]

```

State of karts

```

property players → List[pystk.Player]

```

State of active players

```

property soccer → pystk.Soccer

```

Soccer match info

```

property time → float

```

Game time

class `pystk.Track`

```

update(self: pystk.Track) → None

```

```

property length → float

```

length of the track

```

property path_distance → numpy.ndarray[numpy.float32]

```

Distance down the track of each line segment (float N x 2)

```

property path_nodes → numpy.ndarray[numpy.float32]

```

Center line of the drivable area as line segments of 3d coordinates (float N x 2 x 3)

```

property path_width → numpy.ndarray[numpy.float32]

```

Width of the path segment (float N)

```
class pystk.Player
```

```
    property camera → pystk.Camera  
        Camera parameters of the player
```

```
    property kart → pystk.Kart  
        Kart of the player
```

```
class pystk.Camera
```

```
    class Mode
```

```
        CLOSEUP = 1
```

```
        FALLING = 4
```

```
        LEADER_MODE = 3
```

```
        NORMAL = 0
```

```
        REVERSE = 2
```

```
    property name → str
```

```
    property value → int
```

```
    property aspect → float  
        Aspect ratio
```

```
    property fov → float  
        Field of view
```

```
    property mode → pystk.Camera.Mode  
        Camera mode
```

```
    property projection → readonly_memoryview  
        Projection matrix (float 4x4)
```

```
    property view → readonly_memoryview  
        View matrix (float 4x4)
```

```
class pystk.Item
```

```
    class Type
```

```
        BANANA = 1
```

```
        BONUS_BOX = 0
```

```
        BUBBLEGUM = 4
```

```
        EASTER_EGG = 6
```

```
        NITRO_BIG = 2
```

```
        NITRO_SMALL = 3
```

```
    property name → str
```

```
    property value → int
```

property id → int
Item id compatible with instance data

property location → float3
3D world location of the item

property size → float
Size of the object

property type → *pystk.Item.Type*
Item type

class `pystk.Kart`

property attachment → *pystk.Attachment*
Attachment of kart

property distance_down_track → float
Distance traveled on current lap

property finish_time → float
Time to complete race

property finished_laps → int
Number of laps completed

property front → float3
Front direction of kart 1/2 kart length forward from location

property id → int
Kart id compatible with instance labels

property jumping → bool
Is the kart jumping?

property lap_time → float
Time to completion for last lap

property lives → int
Lives in three strikes battle

property location → float3
3D world location of the kart

property max_steer_angle → float
Maximum steering angle

property name → str
Player name

property overall_distance → float
Overall distance traveled

property player_id → int
Player id

property powerup → *pystk.Powerup*
Powerup collected

property race_result → bool
Did the kart finish the race?

property rotation → Quaternion
Quaternion rotation of the kart

property shield_time → float
Second the shield is up for

property size → float3
Width, height and length of kart

property velocity → float3
Velocity of kart

property wheel_base → float
Wheel base

class pystk.Powerup

class Type

ANVIL = 10

BOWLING = 3

BUBBLEGUM = 1

CAKE = 2

NOTHING = 0

PARACHUTE = 9

PLUNGER = 5

RUBBERBALL = 8

SWATTER = 7

SWITCH = 6

ZIPPER = 4

property name → str

property value → int

property num → int
Number of powerups

property type → *pystk.Powerup.Type*
Powerup type

class pystk.Attachment

class Type

ANVIL = 1

BOMB = 2

BUBBLEGUM_SHIELD = 6

NOTHING = 9

PARACHUTE = 0

```

    SWATTER = 3
    property name → str
    property value → int
    property time_left → float
        Seconds until attachment detaches/explodes
    property type → pystk.Attachment.Type
        Attachment type
class pystk.Soccer

    property ball → pystk.SoccerBall
        Soccer ball information
    property goal_line → List[List[float3[2]][2]]
        Start and end of the goal line for each team
    property score → int[2]
        Score of the soccer match
class pystk.SoccerBall

    property id → int
        Object id of the soccer ball
    property location → float3
        3D world location of the item
    property size → float
        Size of the ball

```

2.5 Logging

PySTK uses a global logging mechanism. You can select one of the log levels below.

```

class pystk.LogLevel
    Global logging level
    debug = 0
    error = 4
    fatal = 5
    info = 2
    property name → str
    property value → int
    verbose = 1
    warn = 3
pystk.set_log_level(arg0: int) → None
    Set the global log level

```

You may also set the log level through an environment variable PYSTK_LOG_LEVEL using a string corresponding to the log level.

Symbols

`__init__()` (*pystk.Race method*), 8

A

`acceleration()` (*pystk.Action property*), 11

`AI_CONTROL` (*pystk.PlayerConfig.Controller attribute*), 9

`animated_characters()` (*pystk.GraphicsConfig property*), 7

`ANVIL` (*pystk.Attachment.Type attribute*), 15

`ANVIL` (*pystk.Powerup.Type attribute*), 15

`aspect()` (*pystk.Camera property*), 13

`attachment()` (*pystk.Kart property*), 14

B

`background` (*pystk.ObjectType attribute*), 11

`ball()` (*pystk.Soccer property*), 16

`BANANA` (*pystk.Item.Type attribute*), 13

`bloom()` (*pystk.GraphicsConfig property*), 7

`BOMB` (*pystk.Attachment.Type attribute*), 15

`bomb` (*pystk.ObjectType attribute*), 11

`BONUS_BOX` (*pystk.Item.Type attribute*), 13

`BOWLING` (*pystk.Powerup.Type attribute*), 15

`brake()` (*pystk.Action property*), 11

`BUBBLEGUM` (*pystk.Item.Type attribute*), 13

`BUBBLEGUM` (*pystk.Powerup.Type attribute*), 15

`BUBBLEGUM_SHIELD` (*pystk.Attachment.Type attribute*), 15

built-in function

`pystk.clean()`, 8

`pystk.init()`, 8

`pystk.is_running()`, 9

`pystk.list_karts()`, 10

`pystk.list_tracks()`, 10

`pystk.set_log_level()`, 16

C

`CAKE` (*pystk.Powerup.Type attribute*), 15

`camera()` (*pystk.Player property*), 13

`CAPTURE_THE_FLAG` (*pystk.RaceConfig.RaceMode attribute*), 10

`CLOSEUP` (*pystk.Camera.Mode attribute*), 13

`config()` (*pystk.Race property*), 9

`controller()` (*pystk.PlayerConfig property*), 9

D

`debug` (*pystk.LogLevel attribute*), 16

`degraded_IBL()` (*pystk.GraphicsConfig property*), 7

`depth()` (*pystk.RenderData property*), 11

`difficulty()` (*pystk.RaceConfig property*), 10

`display_adapter()` (*pystk.GraphicsConfig property*), 7

`distance_down_track()` (*pystk.Kart property*), 14

`dof()` (*pystk.GraphicsConfig property*), 7

`drift()` (*pystk.Action property*), 11

`dynamic_lights()` (*pystk.GraphicsConfig property*), 7

E

`EASTER_EGG` (*pystk.Item.Type attribute*), 13

`error` (*pystk.LogLevel attribute*), 16

F

`FALLING` (*pystk.Camera.Mode attribute*), 13

`fatal` (*pystk.LogLevel attribute*), 16

`ffa()` (*pystk.WorldState property*), 12

`finish_time()` (*pystk.Kart property*), 14

`finished_laps()` (*pystk.Kart property*), 14

`fire()` (*pystk.Action property*), 11

`FOLLOW_LEADER` (*pystk.RaceConfig.RaceMode attribute*), 10

`fov()` (*pystk.Camera property*), 13

`FREE_FOR_ALL` (*pystk.RaceConfig.RaceMode attribute*), 10

`front()` (*pystk.Kart property*), 14

G

`glow()` (*pystk.GraphicsConfig property*), 7

`goal_line()` (*pystk.Soccer property*), 16

H

`hd()` (*pystk.GraphicsConfig static method*), 7

`high_definition_textures()` (*pystk.GraphicsConfig property*), 7

I

`id()` (*pystk.Item property*), 13

id() (*pystk.Kart* property), 14
 id() (*pystk.SoccerBall* property), 16
 image() (*pystk.RenderData* property), 11
 info (*pystk.LogLevel* attribute), 16
 instance() (*pystk.RenderData* property), 11
 items() (*pystk.WorldState* property), 12

J

jumping() (*pystk.Kart* property), 14

K

kart (*pystk.ObjectType* attribute), 11
 kart() (*pystk.Player* property), 13
 kart() (*pystk.PlayerConfig* property), 9
 karts() (*pystk.WorldState* property), 12

L

lap_time() (*pystk.Kart* property), 14
 laps() (*pystk.RaceConfig* property), 10
 ld() (*pystk.GraphicsConfig* static method), 7
 LEADER_MODE (*pystk.Camera.Mode* attribute), 13
 length() (*pystk.Track* property), 12
 light_shaft() (*pystk.GraphicsConfig* property), 7
 lives() (*pystk.Kart* property), 14
 location() (*pystk.Item* property), 14
 location() (*pystk.Kart* property), 14
 location() (*pystk.SoccerBall* property), 16

M

max_steer_angle() (*pystk.Kart* property), 14
 mlaa() (*pystk.GraphicsConfig* property), 7
 mode() (*pystk.Camera* property), 13
 mode() (*pystk.RaceConfig* property), 10
 motionblur() (*pystk.GraphicsConfig* property), 7

N

N (*pystk.ObjectType* attribute), 11
 name() (*pystk.Attachment.Type* property), 16
 name() (*pystk.Camera.Mode* property), 13
 name() (*pystk.Item.Type* property), 13
 name() (*pystk.Kart* property), 14
 name() (*pystk.LogLevel* property), 16
 name() (*pystk.ObjectType* property), 11
 name() (*pystk.PlayerConfig.Controller* property), 9
 name() (*pystk.Powerup.Type* property), 15
 name() (*pystk.RaceConfig.RaceMode* property), 10
 nitro (*pystk.ObjectType* attribute), 11
 nitro() (*pystk.Action* property), 11
 NITRO_BIG (*pystk.Item.Type* attribute), 13
 NITRO_SMALL (*pystk.Item.Type* attribute), 13
 none() (*pystk.GraphicsConfig* static method), 7
 NORMAL (*pystk.Camera.Mode* attribute), 13

NORMAL_RACE (*pystk.RaceConfig.RaceMode* attribute), 10
 NOTHING (*pystk.Attachment.Type* attribute), 15
 NOTHING (*pystk.Powerup.Type* attribute), 15
 num() (*pystk.Powerup* property), 15
 num_kart() (*pystk.RaceConfig* property), 10

O

object (*pystk.ObjectType* attribute), 11
 object_type_shift (*pystk.ObjectType* attribute), 11
 overall_distance() (*pystk.Kart* property), 14

P

PARACHUTE (*pystk.Attachment.Type* attribute), 15
 PARACHUTE (*pystk.Powerup.Type* attribute), 15
 particles_effects() (*pystk.GraphicsConfig* property), 7
 path_distance() (*pystk.Track* property), 12
 path_nodes() (*pystk.Track* property), 12
 path_width() (*pystk.Track* property), 12
 pickup (*pystk.ObjectType* attribute), 12
 PLAYER_CONTROL (*pystk.PlayerConfig.Controller* attribute), 9
 player_id() (*pystk.Kart* property), 14
 players() (*pystk.RaceConfig* property), 10
 players() (*pystk.WorldState* property), 12
 PLUNGER (*pystk.Powerup.Type* attribute), 15
 powerup() (*pystk.Kart* property), 14
 projectile (*pystk.ObjectType* attribute), 12
 projection() (*pystk.Camera* property), 13
 pystk.Action (built-in class), 11
 pystk.Attachment (built-in class), 15
 pystk.Attachment.Type (built-in class), 15
 pystk.Camera (built-in class), 13
 pystk.Camera.Mode (built-in class), 13
 pystk.clean()
 built-in function, 8
 pystk.GraphicsConfig (built-in class), 7
 pystk.init()
 built-in function, 8
 pystk.is_running()
 built-in function, 9
 pystk.Item (built-in class), 13
 pystk.Item.Type (built-in class), 13
 pystk.Kart (built-in class), 14
 pystk.list_karts()
 built-in function, 10
 pystk.list_tracks()
 built-in function, 10
 pystk.LogLevel (built-in class), 16
 pystk.ObjectType (built-in class), 11
 pystk.Player (built-in class), 13
 pystk.PlayerConfig (built-in class), 9
 pystk.PlayerConfig.Controller (built-in class), 9

pystk.Powerup (*built-in class*), 15
 pystk.Powerup.Type (*built-in class*), 15
 pystk.Race (*built-in class*), 8
 pystk.RaceConfig (*built-in class*), 9
 pystk.RaceConfig.RaceMode (*built-in class*), 10
 pystk.RenderData (*built-in class*), 11
 pystk.set_log_level()
 built-in function, 16
 pystk.Soccer (*built-in class*), 16
 pystk.SoccerBall (*built-in class*), 16
 pystk.Track (*built-in class*), 12
 pystk.WorldState (*built-in class*), 12

R

race_result() (*pystk.Kart property*), 14
 render() (*pystk.GraphicsConfig property*), 8
 render_data() (*pystk.Race property*), 9
 rescue() (*pystk.Action property*), 11
 restart() (*pystk.Race method*), 8
 REVERSE (*pystk.Camera.Mode attribute*), 13
 reverse() (*pystk.RaceConfig property*), 10
 rotation() (*pystk.Kart property*), 14
 RUBBERBALL (*pystk.Powerup.Type attribute*), 15

S

score() (*pystk.Soccer property*), 16
 screen_height() (*pystk.GraphicsConfig property*), 8
 screen_width() (*pystk.GraphicsConfig property*), 8
 sd() (*pystk.GraphicsConfig static method*), 7
 seed() (*pystk.RaceConfig property*), 10
 set_ball_location() (*pystk.WorldState static method*), 12
 set_kart_location() (*pystk.WorldState static method*), 12
 shield_time() (*pystk.Kart property*), 15
 size() (*pystk.Item property*), 14
 size() (*pystk.Kart property*), 15
 size() (*pystk.SoccerBall property*), 16
 SOCCER (*pystk.RaceConfig.RaceMode attribute*), 10
 soccer() (*pystk.WorldState property*), 12
 ssao() (*pystk.GraphicsConfig property*), 8
 start() (*pystk.Race method*), 8
 steer() (*pystk.Action property*), 11
 step() (*pystk.Race method*), 9
 step_size() (*pystk.RaceConfig property*), 10
 stop() (*pystk.Race method*), 9
 SWATTER (*pystk.Attachment.Type attribute*), 15
 SWATTER (*pystk.Powerup.Type attribute*), 15
 SWITCH (*pystk.Powerup.Type attribute*), 15

T

team() (*pystk.PlayerConfig property*), 9
 texture_compression() (*pystk.GraphicsConfig property*), 8

THREE_STRIKES (*pystk.RaceConfig.RaceMode attribute*), 10
 time() (*pystk.WorldState property*), 12
 time_left() (*pystk.Attachment property*), 16
 TIME_TRIAL (*pystk.RaceConfig.RaceMode attribute*), 10
 track (*pystk.ObjectType attribute*), 12
 track() (*pystk.RaceConfig property*), 10
 type() (*pystk.Attachment property*), 16
 type() (*pystk.Item property*), 14
 type() (*pystk.Powerup property*), 15

U

unknown (*pystk.ObjectType attribute*), 12
 update() (*pystk.Track method*), 12
 update() (*pystk.WorldState method*), 12

V

value() (*pystk.Attachment.Type property*), 16
 value() (*pystk.Camera.Mode property*), 13
 value() (*pystk.Item.Type property*), 13
 value() (*pystk.LogLevel property*), 16
 value() (*pystk.ObjectType property*), 12
 value() (*pystk.PlayerConfig.Controller property*), 9
 value() (*pystk.Powerup.Type property*), 15
 value() (*pystk.RaceConfig.RaceMode property*), 10
 velocity() (*pystk.Kart property*), 15
 verbose (*pystk.LogLevel attribute*), 16
 view() (*pystk.Camera property*), 13

W

warn (*pystk.LogLevel attribute*), 16
 wheel_base() (*pystk.Kart property*), 15

Z

ZIPPER (*pystk.Powerup.Type attribute*), 15