
pystk Documentation

Release 1.0

Philipp Krähenbühl

Oct 28, 2019

THE BASICS

1	Hardware Requirements	3
2	License	5
2.1	Installation	5
2.2	Quick start	6
2.3	Graphics Setup	6
2.4	Race	8
2.5	Logging	15
	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/philkkr/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. Super-TuxKart 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.

property `animated_characters`
 Animate characters (bool)

property `bloom`
 Enable the bloom effect (bool)

property `degraded_IBL`
 Disable specular IBL (bool)

property `dof`
 Depth of field effect (bool)

property `dynamic_lights`
 Enable dynamic lighting (bool)

property `glow`
 Enable glow around pickup objects (bool)

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

property `high_definition_textures`
 Enable high definition textures 0 / 2

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

property `light_shaft`
 Enable light shafts (bool)

property `m1aa`
 Enable anti-aliasing (bool)

property `motionblur`
 Enable motion blur (bool)

property `particles_effects`
 Particle effect 0 (none) to 2 (full)

property `render_window`
 Show the rendering window (bool)

property `screen_height`
 Height of the rendering surface (int)

property `screen_width`
 Width of the rendering surface (int)

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

property ssao

Enable screen space ambient occlusion (bool)

property texture_compression

Use texture compression (bool)

`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) → None`

property config

The current race configuration (RaceConfig)

property last_action

the last action the agent took (List[Action])

property render_data

rendering data from the last step (List[RenderData])

`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*) \rightarrow None

Stop the race

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()` \rightarrow 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`

Let the player or AI drive, AI ignores `step(action)`

AI_CONTROL

PLAYER_CONTROL

property `controller`

Controller type (`PlayerConfig.Controller`)

property `kart`

Kart type (string), see `list_karts` for a list of kart types

property `team`

Team of the player (int) 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

FOLLOW_LEADER

FREE_FOR_ALL

NORMAL_RACE

SOCCER

THREE_STRIKES

TIME_TRIAL

property difficulty

Skill of AI players 0..2 (int)

property laps

Number of laps the race runs for (int)

property mode

Specify the type of race (RaceMode)

property num_kart

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

property players

List of all agent players (List[PlayerConfig])

property render

Is rendering enabled? (bool)

property reverse

Reverse the track (bool)

property seed

Random seed (int)

property step_size

Game time between different step calls (float)

property track

Track name (str)

`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

Acceleration, normalize to 0..1 (float)

property brake

Hit the brakes (bool)

property drift

Drift while turning (bool)

property fire

Fire the current pickup item (bool)

property nitro

Use nitro (bool)

property rescue

Call the rescue bird (bool)

property steer

Steering angle, normalize to -1..1 (float)

2.4.3 Data

class `pystk.RenderData`

SuperTuxKart rendering output

property depth

Depth image of the kart (memoryview[float] screen_height x screen_width)

property image

Color image of the kart (memoryview[uint8] screen_height x screen_width x 3)

property instance

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

background

bomb

kart

nitro

object

pickup

projectile

track

unknown

2.4.4 Game state

PySTK also exposes the internal state of the game.

class `pystk.WorldState`**property items**

State of items (List[Item])

property karts

State of karts (List[Kart])

property players

State of active players (List[Player])

property time

Game time

update (*self*: *pystk.WorldState*) → None

Update this object with the current world state

class *pystk.Track*

property length

length of the track (float)

property path_distance

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

property path_nodes

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

property path_width

Width of the path segment (float N)

update (*self*: *pystk.Track*) → None

class *pystk.Player*

property camera

Camera parameters of the player (Camera)

property kart

Kart of the player (Kart)

class *pystk.Camera*

class *Mode*

CLOSEUP

FALLING

LEADER_MODE

NORMAL

REVERSE

SIMPLE_REPLAY

property aspect

Aspect ratio (float)

property fov

Field of view (float)

property mode

Camera mode (Camera.Mode)

property projection

Projection matrix (float 4x4)

property view

View matrix (float 4x4)


```
class pystk.Item
```

```
    class Type
```

```
        BONUS_BOX
```

```
        BUBBLEGUM
```

```
        EASTER_EGG
```

```
        NITRO_BIG
```

```
        NITRO_SMALL
```

```
    property id
```

```
        Item id compatible with instance data (int)
```

```
    property location
```

```
        3D world location of the item (float 3)
```

```
    property size
```

```
        Size of the object (float)
```

```
    property type
```

```
        Item type (Item.Type)
```

```
class pystk.Kart
```

```
    property attachment
```

```
        Attachment of kart (Attachment)
```

```
    property distance_down_track
```

```
        Distance traveled on current lap (float)
```

```
    property finish_time
```

```
        Time to complete race (float)
```

```
    property finished_laps
```

```
        Number of laps completed (int)
```

```
    property front
```

```
        Front direction of kart 1/2 kart length forward from location (float 3)
```

```
    property id
```

```
        Kart id compatible with instance labels (int)
```

```
    property jumping
```

```
        Is the kart jumping? (bool)
```

```
    property lap_time
```

```
        Time to completion for last lap (float)
```

```
    property location
```

```
        3D world location of the kart (float 3)
```

```
    property max_steer_angle
```

```
        Maximum steering angle (float)
```

```
    property name
```

```
        Player name (str)
```

property overall_distance

Overall distance traveled (float)

property player_id

Player id (int)

property powerup

Powerup collected (Powerup)

property race_result

Did the kart finish the race? (bool)

property rotation

Quaternion rotation of the kart (float 4)

property shield_time

Second the shield is up for (float)

property size

Width, height and length of kart (float 3)

property velocity

Velocity of kart (float 3)

property wheel_base

Wheel base (float)

class pystk.Powerup

class Type

ANVIL

BOWLING

BUBBLEGUM

CAKE

NOTHING

PARACHUTE

PLUNGER

RUBBERBALL

SWATTER

SWITCH

ZIPPER

property num

Number of powerups (int)

property type

Powerup type (Powerup.Type)

class pystk.Attachment

class Type

ANVIL
BOMB
BUBBLEGUM_SHIELD
NOTHING
PARACHUTE
SWATTER

property time_left
Seconds until attachment detaches/explodes (float)

property type
Attachment type (Attachment.Type)

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

error

fatal

info

verbose

warn

`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*), 10

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

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

`ANVIL` (*pystk.Attachment.Type attribute*), 14

`ANVIL` (*pystk.Powerup.Type attribute*), 14

`aspect()` (*pystk.Camera property*), 12

`attachment()` (*pystk.Kart property*), 13

B

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

`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*), 14

`brake()` (*pystk.Action property*), 10

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

`BUBBLEGUM` (*pystk.Powerup.Type attribute*), 14

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

C

`CAKE` (*pystk.Powerup.Type attribute*), 14

`camera()` (*pystk.Player property*), 12

`CAPTURE_THE_FLAG` (*pystk.RaceConfig.RaceMode attribute*), 9

`CLOSEUP` (*pystk.Camera.Mode attribute*), 12

`config()` (*pystk.Race property*), 8

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

D

`debug` (*pystk.LogLevel attribute*), 15

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

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

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

`distance_down_track()` (*pystk.Kart property*), 13

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

`drift()` (*pystk.Action property*), 10

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

E

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

`error` (*pystk.LogLevel attribute*), 15

F

`FALLING` (*pystk.Camera.Mode attribute*), 12

`fatal` (*pystk.LogLevel attribute*), 15

`finish_time()` (*pystk.Kart property*), 13

`finished_laps()` (*pystk.Kart property*), 13

`fire()` (*pystk.Action property*), 10

`FOLLOW_LEADER` (*pystk.RaceConfig.RaceMode attribute*), 9

`fov()` (*pystk.Camera property*), 12

`FREE_FOR_ALL` (*pystk.RaceConfig.RaceMode attribute*), 9

`front()` (*pystk.Kart property*), 13

G

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

H

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

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

I

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

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

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

`info` (*pystk.LogLevel attribute*), 15

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

`items()` (*pystk.WorldState property*), 11

J

`jumping()` (*pystk.Kart property*), 13

K

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

L

`lap_time()` (*pystk.Kart* property), 13
`laps()` (*pystk.RaceConfig* property), 10
`last_action()` (*pystk.Race* property), 8
`ld()` (*pystk.GraphicsConfig* method), 7
`LEADER_MODE` (*pystk.Camera.Mode* attribute), 12
`length()` (*pystk.Track* property), 12
`light_shaft()` (*pystk.GraphicsConfig* property), 7
`location()` (*pystk.Item* property), 13
`location()` (*pystk.Kart* property), 13

M

`max_steer_angle()` (*pystk.Kart* property), 13
`m1aa()` (*pystk.GraphicsConfig* property), 7
`mode()` (*pystk.Camera* property), 12
`mode()` (*pystk.RaceConfig* property), 10
`motionblur()` (*pystk.GraphicsConfig* property), 7

N

`N` (*pystk.ObjectType* attribute), 11
`name()` (*pystk.Kart* property), 13
`nitro` (*pystk.ObjectType* attribute), 11
`nitro()` (*pystk.Action* property), 10
`NITRO_BIG` (*pystk.Item.Type* attribute), 13
`NITRO_SMALL` (*pystk.Item.Type* attribute), 13
`NORMAL` (*pystk.Camera.Mode* attribute), 12
`NORMAL_RACE` (*pystk.RaceConfig.RaceMode* attribute), 9
`NOTHING` (*pystk.Attachment.Type* attribute), 15
`NOTHING` (*pystk.Powerup.Type* attribute), 14
`num()` (*pystk.Powerup* property), 14
`num_kart()` (*pystk.RaceConfig* property), 10

O

`object` (*pystk.ObjectType* attribute), 11
`object_type_shift` (*pystk.ObjectType* attribute), 11
`overall_distance()` (*pystk.Kart* property), 13

P

`PARACHUTE` (*pystk.Attachment.Type* attribute), 15
`PARACHUTE` (*pystk.Powerup.Type* attribute), 14
`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), 11

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

`player_id()` (*pystk.Kart* property), 14
`players()` (*pystk.RaceConfig* property), 10
`players()` (*pystk.WorldState* property), 11
`PLUNGER` (*pystk.Powerup.Type* attribute), 14
`powerup()` (*pystk.Kart* property), 14
`projectile` (*pystk.ObjectType* attribute), 11
`projection()` (*pystk.Camera* property), 12
`pystk.Action` (built-in class), 10
`pystk.Attachment` (built-in class), 14
`pystk.Attachment.Type` (built-in class), 14
`pystk.Camera` (built-in class), 12
`pystk.Camera.Mode` (built-in class), 12
`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), 12
`pystk.Item.Type` (built-in class), 13
`pystk.Kart` (built-in class), 13
`pystk.list_karts()` (built-in function), 10
`pystk.list_tracks()` (built-in function), 10
`pystk.LogLevel` (built-in class), 15
`pystk.ObjectType` (built-in class), 11
`pystk.Player` (built-in class), 12
`pystk.PlayerConfig` (built-in class), 9
`pystk.PlayerConfig.Controller` (built-in class), 9
`pystk.Powerup` (built-in class), 14
`pystk.Powerup.Type` (built-in class), 14
`pystk.Race` (built-in class), 8
`pystk.RaceConfig` (built-in class), 9
`pystk.RaceConfig.RaceMode` (built-in class), 9
`pystk.RenderData` (built-in class), 11
`pystk.set_log_level()` (built-in function), 15
`pystk.Track` (built-in class), 12
`pystk.WorldState` (built-in class), 11

R

`race_result()` (*pystk.Kart* property), 14
`render()` (*pystk.RaceConfig* property), 10
`render_data()` (*pystk.Race* property), 8
`render_window()` (*pystk.GraphicsConfig* property), 7
`rescue()` (*pystk.Action* property), 10
`restart()` (*pystk.Race* method), 8
`REVERSE` (*pystk.Camera.Mode* attribute), 12
`reverse()` (*pystk.RaceConfig* property), 10
`rotation()` (*pystk.Kart* property), 14
`RUBBERBALL` (*pystk.Powerup.Type* attribute), 14

S

`screen_height()` (*pystk.GraphicsConfig* property),

7

`screen_width()` (*pystk.GraphicsConfig* property), 7
`sd()` (*pystk.GraphicsConfig* method), 7
`seed()` (*pystk.RaceConfig* property), 10
`shield_time()` (*pystk.Kart* property), 14
`SIMPLE_REPLAY` (*pystk.Camera.Mode* attribute), 12
`size()` (*pystk.Item* property), 13
`size()` (*pystk.Kart* property), 14
`SOCCER` (*pystk.RaceConfig.RaceMode* attribute), 9
`ssao()` (*pystk.GraphicsConfig* property), 7
`start()` (*pystk.Race* method), 8
`steer()` (*pystk.Action* property), 10
`step()` (*pystk.Race* method), 8
`step_size()` (*pystk.RaceConfig* property), 10
`stop()` (*pystk.Race* method), 9
`SWATTER` (*pystk.Attachment.Type* attribute), 15
`SWATTER` (*pystk.Powerup.Type* attribute), 14
`SWITCH` (*pystk.Powerup.Type* attribute), 14

T

`team()` (*pystk.PlayerConfig* property), 9
`texture_compression()` (*pystk.GraphicsConfig* property), 8
`THREE_STRIKES` (*pystk.RaceConfig.RaceMode* attribute), 9
`time()` (*pystk.WorldState* property), 11
`time_left()` (*pystk.Attachment* property), 15
`TIME_TRIAL` (*pystk.RaceConfig.RaceMode* attribute), 9
`track` (*pystk.ObjectType* attribute), 11
`track()` (*pystk.RaceConfig* property), 10
`type()` (*pystk.Attachment* property), 15
`type()` (*pystk.Item* property), 13
`type()` (*pystk.Powerup* property), 14

U

`unknown` (*pystk.ObjectType* attribute), 11
`update()` (*pystk.Track* method), 12
`update()` (*pystk.WorldState* method), 12

V

`velocity()` (*pystk.Kart* property), 14
`verbose` (*pystk.LogLevel* attribute), 15
`view()` (*pystk.Camera* property), 12

W

`warn` (*pystk.LogLevel* attribute), 15
`wheel_base()` (*pystk.Kart* property), 14

Z

`ZIPPER` (*pystk.Powerup.Type* attribute), 14