Panda Camp User Manual

Introduction

Camp philosophy; how should someone use this code.

Panda 3D

Introduce Panda 3D and the reactive engine.

Math

Explain the use of math in 3D environments and the engine.

pi - A value used to describe measures in radians.

cos(a) - A function that oscillates between -1 and 1 every 2pi units of time, starting at 0 (a = 0).

sin(a) - A function that oscillates between -1 and 1 every 2pi units of time, starting at 1 (a = 0).

integral(a) - This changes a speed (a) into a position.

The following sets the position of a panda using a speed while oscillating the HPR with sin and cos

```
    p0 = P3(0, 0, 0)
    speed = P3(1, 0, 1)
    pos = p0 + integral(speed)
    panda(position = pos, hpr = HPR(sin(time), cos(time), 0))
```

Python

What explanation should we give?

range(a, b) - Returns a list of numbers between a lower bound (a) and upper bound (b). If only one parameter is given then 0 is the lower or upper bound.

randomRange(a, b) - Generates a random number between a lower bound (a) and an upper bound (b).

random01() - Generates a random number between 0 and 1.

for - Allows you to do something more than once.

The following uses a for loop to generate 10 random numbers between 0 and 100

```
    for i in range(10):
        print randomRange(0,100)
```

Loops can be nested for more complex tasks. The following nests two loops to create a ten by ten grid of pandas

```
    for i in range(10):
        for j in range(10):
            panda(position = P3(i, 0, j))
```

if - A condition that, if met, causes something to happen.

elif - Another condition following an if or elif that, if met, causes something else to happen.

else - Causes something to happen only if the previous if and/or elif(s) aren't met.

```
    if someBoolean: // this is the same as "someBoolean == True"
        // do something
    elif !someOtherBoolean: // this is the same as "someOtherBoolean != True"
        // do something else
    else:
        // if neither of the previous clauses hold
```

Discuss type casting and inference as well? What about recursion?
A function is how programmers are able to reuse code - do the same thing more than once, perhaps in slightly different ways. Here is a function that creates a panda

```
    def f(p):
        panda(position = p)
```

Here the argument p to the function f is a P3. For further discussion of the use of functions see the Events and Reactions section.

Reactive

Briefly describe what reactive code is in the context of the engine. The following reactive variables are available

var(a) - A reactive variable with an initial value a. This has the following reactive parameters

.add(a) - Adds a to the variable.

P3(x, y, z) - Defines a location in 3-space.

P2(x, y) - Defines a location in 2-space (i.e. the surface of the screen)

HPR(h, p, r) - Determines the spin of an object using heading, pitch, and role; defines the objects orientation in 3-space

Color(r, g, b) - Defines color using the red (r), green (g), and blue (b) channels with values between 0 and 1 (see the "Colors" section for a list of predefined colors)

time - A linear unit of measurement that begins when the program starts. This is variously used for movement and other reactive behavior.

locelTime - The time since an object has been created.

Non-Reactive

Briefly describe what non-reactive code is in the context of the engine. Is there any worth mention? Is this section necessary?

Interface

Describe the window that everything will run in and discuss the reactive stuffs that can be added.

text(a) - Prints reactive and non-reactive types to the screen. This includes strings, floats, and P3s.

format(s, a) - This is used to combine strings (s) and other reactive and non-reactive variables (a) into a reactive form legible by text.

The following is an example of how to use text and format

```
    text(format("You have scored %i in %f seconds", score, time))
```

Here %i and %f are conversion types, respectively representing the variables score (an int) and time (a float). The following conversion types are available

i - Signed integer.

x - Unsigned hexadecimal (lowercase).

X - Unsigned hexadecimal (uppercase).

f - Floating point (decimal format).

c - Single character (an integer or a single character string).

r - Converts any python object using repr().

s - Converts any python object using str().

Sliders are used to change the value a variable via the interface. The following sliders are available

slider() - Creates a slider that returns a floating point value.

sliderP3() - Creates a slider that returns P3 information.

sliderHPR() - Creates a slider that returns HPR information.

sliderColor() - Creates four sliders that set a color.

Sliders have the following parameters

label (string, default value = None) - Text that will appear to the left of a slider.

min (float, default value = 0) - The minimum value of the slider (when it's all the way left).

max (float, default value = 1) - The maximum value of the slider (when it's all the way right).

init (float, default value = None) - The initial value of the slider.

size (float, default value = 1) - The size of the slider.

position (P3, default value = None) - Position on screen.

Here a slider is used to control the x-value of a panda's position

```
x = slider(label = "x-value", min = -3, max = 3, init = 0)
panda(position = P3(x, 0, 0))
```

Here a color slider is used to change the background color of the world

```
color = sliderColor()
world.color = color
```

check-boxes?

anything else?

Models

A model is a predefined object that appears on the screen whose representation is obtained from a 3D-model in an .eeg file. By default the model is on the xy-plane facing in the y direction - towards the camera. Models have the following reactive parameters

position (P3, default value = P3(0, 0, 0)) - Location in 3-space.

hpr (HPR, default value = HPR(0, 0, 0)) - Orientation in 3-space.

scale (scalar, default value = 1 - 1 unit cube) - Relative scale.

color (Color, defule = None) - Blends a color with a models texture.

These parameters may be given values during initialization as follows

```
panda(position = P3(1, -2, 3.7), hpr = HPR(20, pi, time), scale = 0.76, color = Color(0, 0.3,
    random01()))
```

They may also be given values after initialization by placing the model in a variable as follows

```
p = panda()
p.position = P3(-3, 4, randomRange(-2, 4))
p.hpr = HPR(cos(time), sin(time), 0)
p.scale = sin(time)/3 + 1.2
p.Color(0.45, 0.65, 0.23)
```

The value of these parameters may be retrieved at any time in a similar fashion. The following places the following model's hpr on the screen

```
text(p.hpr)
```

Models have the following non-reactive variables

cRadius (scalar) - The radius of the hypothetical cylinder that contains the model.

cFloor (scalar) - The position on the z-axis of the bottom of hypothetical cylinder that contains the model.

cTop (scalar) - The position on the z-axis of the top of hypothetical cylinder that contains the model.

cType (Type) - The model's type.

duration (scalar, defualt = None) - How long the model will remain on the screen. If None the model will not automatically disappear.

collection (Collection, default = None) - The collection the model belongs to.

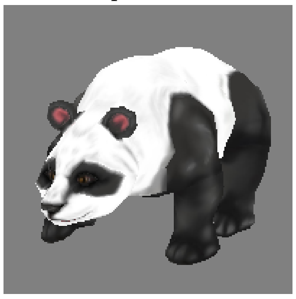
A new variable can be added to a model on the fly as follows

```
    p = panda()
    p.cute = True
```

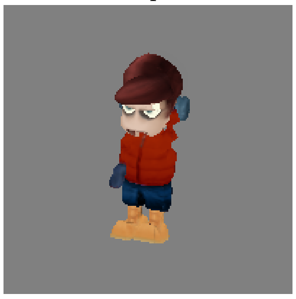
Also, the following function is available for all models

pointForward(model) - Points the model in the -y direction - the same as the camera by default - therefor forward.

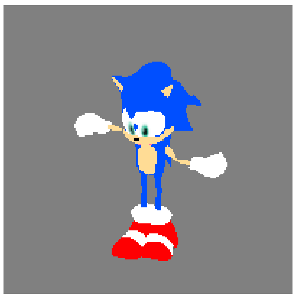
panda



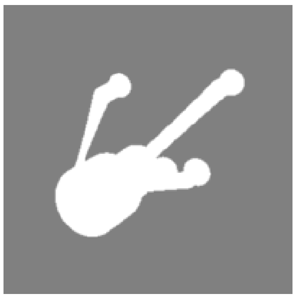
ralph



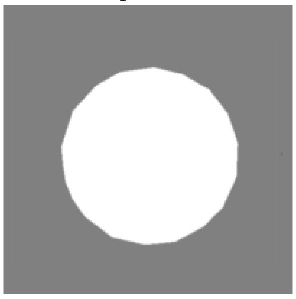
sonic



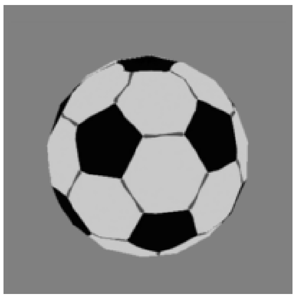
bender



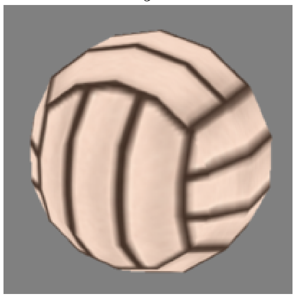
sphere



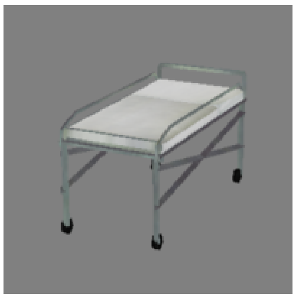
soccerBall



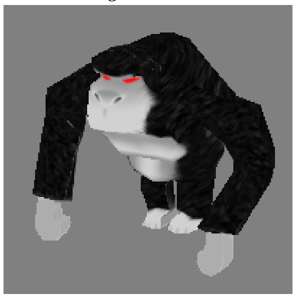
volleyBall



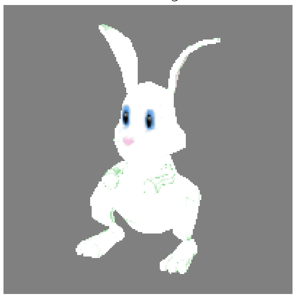
stretcher



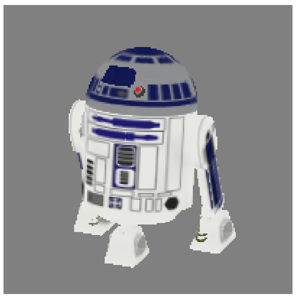
gorilla



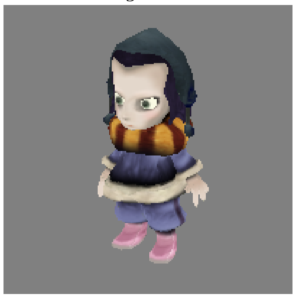
bunny



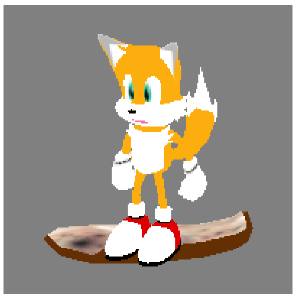
r2d2



girl



tails

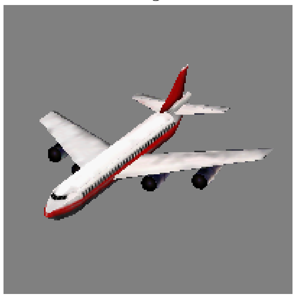


truck

ford

jeep

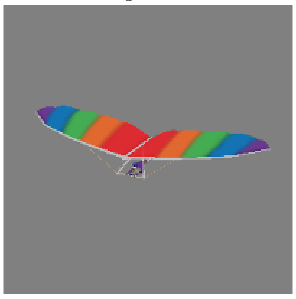
boeing707



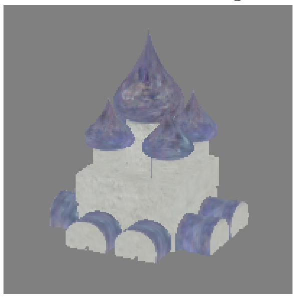
blimp

spaceship

hangGlider



russianBuilding



Rectangle

Rectangles are geometric objects that are made using three points: lower-left, lower-right, and upper-left. These are P3 points and are the only required parameters of the rectangle object. Rectangles have the following optional properties

texture (Color/texture, default value = None) - Used to set the color or the texture of the rectangle.

The color is set as follows

```
    rectangle(P3(0, 0, 0), P3(1, 0, 0), P3(0, 1, 0), texture = purple)
```

A texture is an image and may be set like this

```
    rectangle(P3(0, 0, 0), P3(1, 0, 0), P3(0, 1, 0), texture = "image.jpg")
```

side2 (Color/texture, defualt value = None) - Used to set the color or the texture of the back side of the rectangle. If left blank this will be the same color or texture as the front of the rectangle.

Like models, rectangles have the following parameters (see the Models section for the descriptions)

position

hpr

size - This is synonymous to the models' scale parameter.

color

Rectangles may be used to create various geometric solids as well.

blastPicture - Takes a picture and slices it into smaller rectangles. Each of these rectangles has the following attributes

.x (float) - It's x position.

.y (float) - It's y position.

.location (P3) - It's location in 3-space.

If a picture is blasted and each fragment's position is set to it's .location, the picture will be reassembled. The following does just that and then spins the fragments

```
    fragments = blastPicture("image.jpg", 5, 5)

    for p in fragments:
        p.position = p.location
        p.hpr = HPR(time*random01(), 0, 0)
```

wheel - Takes an arbitrary number of images as it's arguments and then uses them to create a wheel.

cube - Takes six images as it's arguments and places one on each side of a cube.

tetrahedron - Takes four images as it's arguments and places one on each side of a tetrahedron.

The following creates a wheel with five images

```
    photoWheel(["image1.jpg", "image2.jpg", "image3.jpg", "image4.jpg", "image5.jpg"])
```

Movement

Describe movement in a 3D environment (creating paths?)

step(a) - A function that changes from 0 to 1 when the input (a) goes from positive to negative.

The following moves a panda after 2 seconds

```
    panda(position = P3(step(time-2), 0, 0))
```

smoothStep - The same as step but smoothly steps rather than instantly stepping.

More complex paths can be created using the following interpolation functions

at(a) - Puts the model at a given location (a - a P3).

to(a, b) - Moves the model to a location (b) over a given amount of time (a).

itime(a) - Uses the current time to select a place along the current path (a).

The following creates a path and moves a panda along it

```
    p = panda()
    path = at(P3(0, 0, 0)) + to(1, P3(1, 0, 0)) + to(1, P3(-1, 0, 1))
    p.position = itime(path)
```

Controls

The following reactive values are available to read user input

mouse (P2) - Returns the position of the mouse with respect to the runtime window with values between -1 and 1, (0, 0) being the center.

getX(mouse) - Returns a floating point of the mouse's X value.

getY(mouse) - Returns a floating point of the mouse's Y value.

The mouse can be used to control the position of a panda as follows

```
    panda(position = P3(getX(mouse), 0, getY(mouse)))
```

lbutton (boolean) - Whether the left-mouse-button is pressed or not.

rbutton (boolean) - Whether the right-mouse-button is pressed or not.

An event is something that happens as the program runs, like mouse clicks and keys being pressed. The following events are available

lbp - Left-mouse-button pressed.

rbp - Right-mouse-button pressed.

rbr - Right-mouse-button released.

lbr - Left-mouse-button released.

These, and other keys, are given values using the following functions

key(a, b) - Assigns the value b to the name a when a is pressed.

keyUp(a, b) - Assigns the value b to the name a when a is released.

The following are special key name

"f"+"1-12" - These are the function keys, i.e. "f1", "f2", "f3", ... , "f12".

"scroll_lock"

"backspace"

"insert"

"home"

"page_up"

"num_lock"

"tab"

"delete"

"end"

"page_down"

"caps_lock"

"enter"

"arrow_left"

"arrow_up"

"arrow_down"

"arrow_right"

"shift" - Either shift key.

"lshift" - Left-shift.

"rshift" - Right-shift.

"control" - Either control.

"lcontrol" - Left-control.

"rcontrol" - Right-control.

"alt" - Either alt.

"lalt" - Left-alt.

"ralt" - Right-alt.

"space"

An event can happen when more than one key is pressed. An event can be composed of multiple keys using the + operator as follows

```
    p = panda()
    v = hold(P3(0, 0, 0), key("left-arrow", P3(-1, 0, 0)) + key("right-arrow", P3(1, 0, 0)))
    p.position = P3(0, 0, -2) + integral(v)
```

Events and Reactions

Discuss how events work (at a certain point in time) and why they're useful.

timeIs(a) - Triggers when the time variable reaches a.

now(a) - Takes the value of a at that moment.

hold(a, b) - Holds the value a after the event b occurs.

alarm(a) - Generates an event at a given time step (a).

happen(a, b) - Turns a boolean a into an event b.

Discuss how reactions work and why they're useful. Attach a reaction to a model by giving the triggering event and the name of the reaction function.

react(a, b) - Calls the reaction b when the event a happens.

react1(a, b) - Same a react but it's reaction is the only executed code after the event.

Here is an example of using react with an alarm

```
a = alarm(step = 0.8)
react(a, doSomething)
```

when(a, b) - Same as react but happens when a condition a (true / false) is met.

when1(a, b) - The same as when but it's reaction is the only executed code after the event.

Particle Effects

Particle effects are created much like models, having the same attributes and parameters. The following particle effects are available.

intervalRings

likeFountainWater

shakenbSparkles

warpSpeed

heavySnow

lightSnow

explosions

fireWork

fireWorks

explosion

warpFace

fireish

Camera

The camera is how the scene is viewed. By default the camera is on the xy-plane pointing in the -y direction - towards the models default position.

mouseControlCamera(camera) - Controls the camera using the mouse.

.rod(a) - This places the camera on a hypothetical rod behind an object (a), causing it to follow the object wherever it may go. It has the following parameters

distance (float, default value = 3) - Distance of the camera from the object.

height (float, default value = 0.5) - Height of camera off the xy-place.

Light

Describe the emulation of light in virtual worlds. When no light source is specified, the world is filled with ambient light - light that comes from no particular direction. By adding a light source, the ambient light is turned off. The following light sources are available

pointLight - Creates light from a single point. This has attributes similar to the models

- position (P3)
- hpr (HPR)
- color (Color)

ambientLight - This creates a light that comes from no particular direction.

directionalLight - Illuminates surfaces facing in the same direction identically. There are the following parameters

um...

Maps

Describe maps and the various reactive and non-reactive components.

Sound

Sound may be played in using the following methods

Sound(s) - This creates a sound object that plays the sound file s. This object also has the following parameters

- loopCount (int, default value = 1) - How many times the file will play.
- volume (float, defauult value = 0.5) - The volume of the sound, 0 being mute and 1 being full.

play(s) - Plays the sound file s once.

Useful Stuff

This is where all the extra stuff goes.

resetWorld() - Removes everything from the world, interface objects included.

collection - A reactive group of objects?

exitScene() - A reaction the removes a model from the world.

HPRtpP3(a) - Takes HPR a and converts it to a P3.

P3toHPR(a) - Takes P3 a and converts it to an HPR.

P3C(a, b) - Creates a P3 from cylindrical coordinates (angle a and length b)

Colors

⬛ black

🟦 blue

🟦 darkBlue

🟦 lightBlue

⬛ grey

⬛ gray

⬜ lightGray

⬛ darkGray

🟩 limeGreen

🟩 green

🟩 darkGreen

🟥 hotpink

🟥 red

🟥 darkRed

⬜ white

🟨 yellow

🟨 gold

⬜ silver

🟦 navyBlue

🟪 purple

🟪 deepPurple

🟪 violet

🟫 brown

🟧 orange

🟦 teal

🟦 cyan

🟩 aquamarine

🟦 slateGray

🟪 deepPurple

🟧 bloodOrange

🟪 fuchsia

🟫 tan

🟧 lightTan

🟩 olive

🟩 springGreen

🟥 coral

🟥 salmon

🟪 lavender

Examples

Give some neat examples that draw on the content of the manual.
Here's a demonstration of a code example from a file.

Code Example Name

```
# demo.py

from Panda import *

for i in range(10):
    for j in range(10):
        panda(position = P3(i, 0, j))
```