Blender-VR Temple Source

Release 1.0

Blender-VR Team

September 14, 2015

Contents

The documentation of the Blender-VR Temple demo project

CHAPTER 1

Source Code

1.1 src package

1.1.1 Submodules

Artificial Intelligence

Control the enemies behaviour (bats, ghosts, pendulum)

src.ai.**BL_EnemyArmatureObject**
alias of BL_ArmatureObject

class src.ai.**Base**(*parent*)
Bases: src.base.Base

bats

ghosts

loop()

pendulums

spawnEnemies()
populate enemies

trailSeek()
see if we are to be activated

trailSeeker(*ob*, *controller*, *actuator*)
Store the object we use to evaluate the trail for enemy spawning

class src.ai.**Bat**(*scene*, *obj*, *target*, *speed*, *events*, *logger*)
Bases: src.ai.FlyingEnemy

enemy = 'BAT'

ray_filter = 'bat'

class src.ai.**Enemy**(*speed*, *events*, *logger*)
Bases: builtins.object

addObject(*scene*, *object_name*, *object_origin*)
Spawn a new object in the game

attack(*origin*)
Check if enemy is close enough to eat

attack_distance_squared = 0.01

classmethod calculateId()
Generates a new id based on the number of added instances

changeState()
Called when the object changes to a relevant state, called from Logic Bricks

end()
End the object, called from Logic Bricks

evade_distance_squared = 100.0

static getState(*state*)
Return the bitwise flag corresponding to this state

classmethod hit(*camera*, *origin*, *direction*)
Try to hit an enemy from this origin at this direction If succeds, send a message to end the object

init()
Initialize the object, called from Logic Bricks

instances = 0

kill()
Send message to eliminate the object It is called when we hit the enemy or the enemy hits us

ray_filter = ''

setSound(*sound*)
Setup OSC sound engine, called from sound.py

sound_source
Return the object to use as reference for the sound origin

subject
Message subject to use with the Message sensor to end the object

class src.ai.**FlyingEnemy**(*name*, *scene*, *obj*, *target*, *speed*, *events*, *logger*)
Bases: src.ai.Enemy

class src.ai.**Ghost**(*scene*, *obj*, *target*, *speed*, *events*, *logger*)
Bases: src.ai.FlyingEnemy

activation_distance = 30.0

attack_distance_squared = 0.25

enemy = 'GHOST'

ray_filter = 'ghost'

src.ai.**KX_EnemyGameObject**
alias of KX_GameObject

class src.ai.**Pendulum**(*scene*, *obj*, *speed*, *events*, *logger*)
Bases: src.ai.Enemy

attack()
Called from Logic Brick callback We are already hitting the player

end()
End the object, called from Logic Bricks

enemy = 'PENDULUM'

evade(*origin*)
Check if enemy is too far

ray_filter = 'pendulum'

class src.ai.**Seeker**(*ob*, *controller*, *actuator*)
Bases: builtins.object

getOrientation(*frame*, *callback*, *user_data=None*)
Get orientation matrix of the animation at a given frame

Parameters

- **frame** (*int*) – animation frame
- **callback** (*function(mathutils.Matrix, user_data)*) – callback function
- **user_data** (*Object*) – user data passed back to callback function

getPosition(*frame*, *callback*, *user_data=None*)
Get position of the animation at a given frame

Parameters

- **frame** (*int*) – animation frame
- **callback** (*function(mathutils.Vector, user_data)*) – callback function
- **user_data** (*Object*) – user data passed back to callback function

getTransform(*frame*, *callback*, *user_data=None*)
Get position and orientation matrix of the animation at a given frame

Parameters

- **frame** (*int*) – animation frame
- **callback** (*function(((mathutils.Vector, mathutils.Matrix), user_data)*) – callback function
- **user_data** (*Object*) – user data passed back to callback function

loop()
activate the actuator to return any stacked query

src.ai.**attacked**(*cont*)
Called from Logic Bricks upon collision with enemy

Only pendulum collides this way, the other objects Use the steering actuator which is incompatible with physics sensors

src.ai.**changeState**(*cont*)
Called from Logic Bricks upon change to relevant states (e.g., start chasing, or end object)

src.ai.**trailSeeking**(*cont*)
activate trail seeking actuator

Base

Basic class for all inherited classes.

class src.base.**Base**(*parent*)
Bases: builtins.object

loop()

class src.base.**Pipe**(*parent*, *name*)
Bases: builtins.object

Pipe commands from the base class (children) to the events module

e.g., in io.py you can do: self.setFlashlightMode(power=True) and this will call events.setFlashlightMode(power=True)

Debug

This module is intended to run and debug the demo without BlenderVR

class src.debug.**Base**(*parent*)
Bases: src.base.Base

loop()
Run once per frame called from a callback

Events Manager

Centralize the communication between the game elements, the navigation devices, the input systems and the sound engine.

class src.events.**Base**(*parent*)
Bases: src.base.Base

evadeEnemy(*enemy*)
Enemy got too distant, it can go away

gameOver()
The game time is over

hitByEnemy(*enemy*)
Enemy got the upper hand

hitEnemy(*enemy*)
Enemy got hit, congratulations

setFlashlightMode(*power=True*)
Activate or Deactivate the flashlight

setSonarMode(*power=True*)
Activate or Deactivate the sonar

spawnEnemy(*enemy*)
A new enemy got spawned

startLap()
Start a new lap

throwRock()
Throw a rock (to hit a pendulum)

Input/Output

Takes the three different inputs, process the data and call the corresponding event. It also handles head transformation/navigation.

class src.io.**Base**(*parent*)
Bases: src.base.Base

enableHeadTrack(*user*)
Use Headtrack (instead of mouse) to control the scene Called from BlenderVR processor file

Parameters **user** – BlenderVR User

flashlightButton()
Flashlight button was pressed

head_direction
Get the direction of the player's head

Return type mathutils.Vector (normalized)

head_orientation
Get the direction of the player's head

Return type mathutils.Quaternion

head_position
Get the direction of the player's head

Return type mathutils.Vector

is_flashlight

is_sonar

loop()

rockButton()
Rock button was pressed

sonarButton()
Sonar button was pressed

Logger

Handles all printing/logging operations

class src.logger.**Base**
Bases: builtins.object

class src.logger.**Logger**
Bases: builtins.object

class src.logger.**Print**(*function*)
Bases: builtins.object

class src.logger.**Printer**
Bases: builtins.object

Scoring System

Keep track of player's progress, score and final results

class src.score.**Base**(*parent*)
Bases: src.base.Base

evade(*enemy*)
Enemy goes away

hit(*enemy*)
Hit (kill) an enemy

hitBy(*enemy*)
Got attacked by an enemy

spawn(*enemy*)
Add a new enemy in the game

Sound

Connected with the game events systems it deals directly with the OSC server It also provides a phantom layer, which mimics OSC with local BGE sound resources.

class src.sound.**AudaspaceSoundEngine**(*logger*)
Bases: src.sound.SoundEngine

class src.sound.**AudaspaceSoundObject**(*engine*, *kx_object*)
Bases: builtins.object

play(*sound*, *loop=False*, *volume=0.5*)
Load and play sound file

class src.sound.**Base**(*parent*)
Bases: src.base.Base

setOSCUser(*user*)
Specify the OSC User to use

setVolumeHigh()
High volume, sonar is on

setVolumeLow()
Low volume, flashlight is on

setVolumeNormal()
Normal volume, initial

class src.sound.**Bat**(*engine*, *sound_source*, *force_fallback=False*)
Bases: src.sound.Enemy

sound_end = 'bat_end.wav'

sound_init = 'bat.wav'

class src.sound.**Enemy**(*engine*, *sound_source*, *sound_init*, *sound_end*, *force_fallback*)
Bases: builtins.object

audio_folder = '//../audio/'

osc = None

playEnd()
Play sound for when the object ends (e.g., is hit by rock)

playInit()
Play sound for when the object is active (e.g., flying)

sound_end

sound_init

class src.sound.**Ghost**(*engine*, *sound_source*, *force_fallback=False*)
Bases: src.sound.Enemy

sound_end = 'ghost_end.wav'

sound_init = 'ghost.wav'

class src.sound.**OSCSoundEngine**(*osc*, *logger*)
Bases: src.sound.SoundEngine

getUser()
called from the OSC objects

setUser(*user*)
called from the processor file

class src.sound.**OSCSoundObject**(*osc*, *engine*, *kx_object*)
Bases: builtins.object

play(*sound*, *loop=False*, *volume=0.5*)
Load and play sound file

Parameters **sound** (*str*) – The sound file filepath

class src.sound.**Pendulum**(*engine*, *sound_source*, *force_fallback=False*)
Bases: src.sound.Enemy

sound_end = 'pendulum_end.wav'

sound_init = 'pendulum.wav'

class src.sound.**SoundEngine**(*logger*, *volumes*)
Bases: builtins.object

setVolumeHigh()
Set the highest volume level

setVolumeLow()
Set the lowest volume level

setVolumeNormal()
Set the regular (initial) volume level

Timeline

Control the flux of the game, and the time related events (spawn of enemies, game end, game start, ...).

class src.timeline.**Base**(*parent*)
Bases: src.base.Base

loop()

1.1.2 Module contents

Blender-VR Temple Source

class src.**Temple**

Bases: builtins.object

ai

bumpSpeed()

Increase global speed

debug

events

io

is_debug

logger

run()

Run once per frame, called from the processor file

score

sound

speed

timeline

src.**main**()

CHAPTER 2

Indices and tables

- *genindex*
- *modindex*
- *search*

S

K

L

M

O

P

R

S

T