2015-012
felix caffier
framework2d starter kit quick documentation

ABSTRACT

this STE contains an easy solution for 2D/2.5D style side scroller games. a demo with the standard shiva assets is included, so you know where things go and how they are supposed to behave.

LICENSE

FEATURES

- Keyboard and Controller support
- framerate independence
- wall collision detection
- physics based movement
- tweakables: mass, drag, friction, air resistance, gravity, movement/jump forces and limits
- dynamic character model support

NOT INCLUDED / CAVEATS

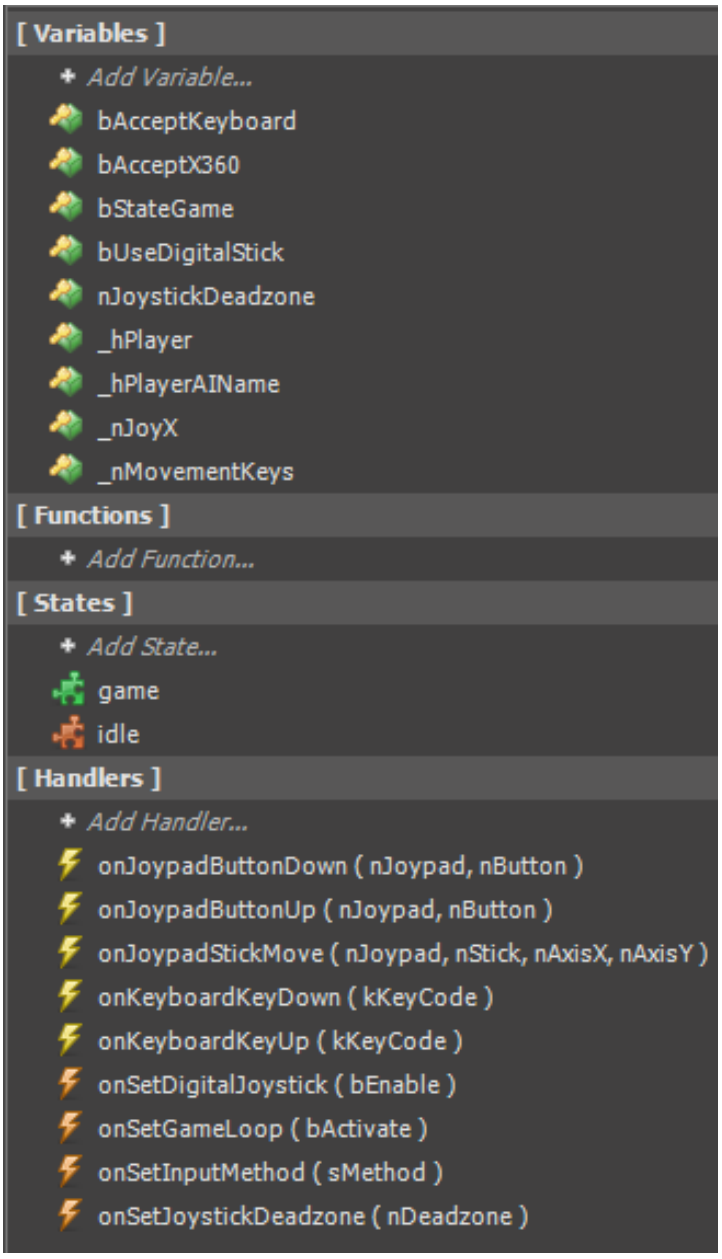
- Camera system not included
- level changing not included
- player model should face -Z = forward
- games HAVE to move along the Z axis
- beware of the guardbox - check framework2d_player.initDynamics()
- player model should have its root at 0,0,0
- player model should have its animation (esp. walk cycle) in place
- would be nice to have at least a jumping + falling animation in addition to idle + walking
- beware of the animation setup code in the _playercharacter AI model

PROJECT STRUCTURE

1. Game with framework2d_input AI (1) - input handler
2. Game with main AI (2) -> loads scene and sets scene options (input model, gravity, ...)
3. scene contains “player” helper which dynamically generates everything there is to the player physics and interaction based on the framework2d_player AI
4. framework2d_player initializes a 3D model for the player, which is controlled by framework2d_playercharacter
5. all AI member variables you can change have a standard name, but those you cannot or should not changed are prefixed by an underscore, like _nJoyX

framework2d_input

- standard handlers for joystick and keyboard input. change for different key assignment
- custom handlers to set the member variables



[Variables]
+ Add Variable...
bAcceptKeyboard
bAcceptX360
bStateGame
bUseDigitalStick
nJoystickDeadzone
_hPlayer
_hPlayerAIName
_nJoyX
_nMovementKeys

[Functions]
+ Add Function...

[States]
+ Add State...
game
idle

[Handlers]
+ Add Handler...
onJoypadButtonDown (nJoypad, nButton)
onJoypadButtonUp (nJoypad, nButton)
onJoypadStickMove (nJoypad, nStick, nAxisX, nAxisY)
onKeyboardKeyDown (kKeyCode)
onKeyboardKeyUp (kKeyCode)
onSetDigitalJoystick (bEnable)
onSetGameLoop (bActivate)
onSetInputMethod (sMethod)
onSetJoystickDeadzone (nDeadzone)

- bAcceptKeyboard/X360: true to use keyboard/Xbox360 controller input

- bUseDigitalStick: <50% stick is 0, >50% stick is 1 - instead of analog

- nJoystickDeadzone: to avoid bogus input, 0.2 recommended

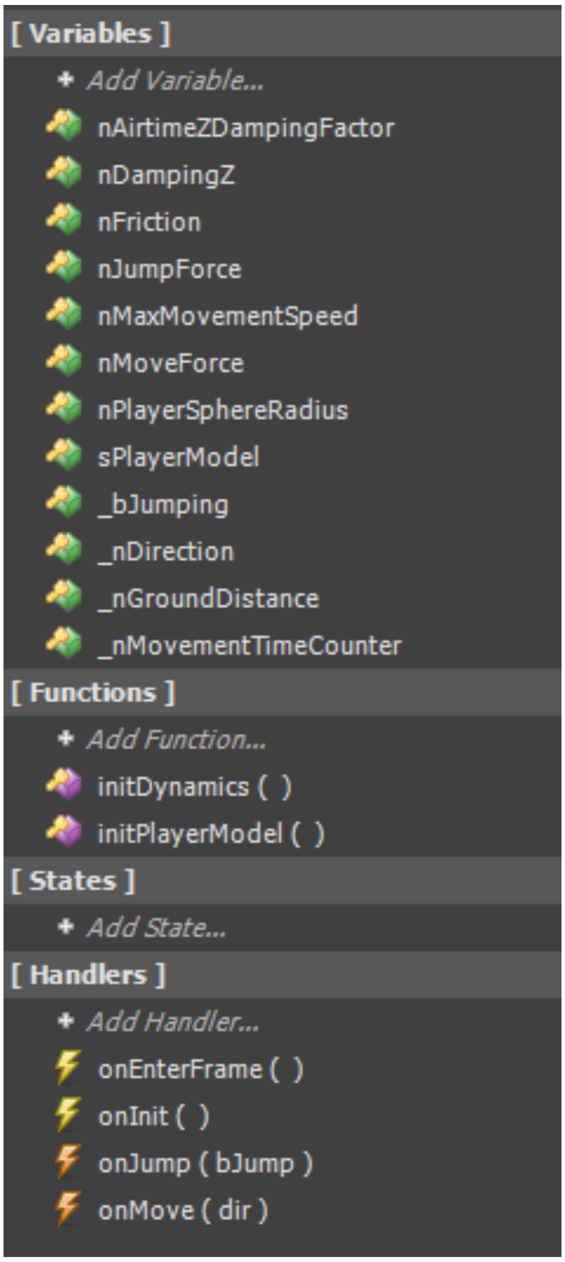
- 2 states: one for game (accept input), one for idle (do nothing)

- standard handlers for input

- handlers to set member variables, call onInit of your game

framework2d_player

- player helper: creates a physics ball and translates inputs to movement and actions
- set the member variables to your liking, then apply the AI to a helper, then drop the helper into your scene, so the setup is always the same


[Variables]
+ Add Variable...
nAirtimeZDampingFactor
nDampingZ
nFriction
nJumpForce
nMaxMovementSpeed
nMoveForce
nPlayerSphereRadius
sPlayerModel
_bJumping
_nDirection
_nGroundDistance
_nMovementTimeCounter
[Functions]
+ Add Function...
initDynamics ( )
initPlayerModel ( )
[States]
+ Add State...
[Handlers]
+ Add Handler...
onEnterFrame ( )
onInit ( )
onJump (bJump)
onMove (dir)

- Airtime Damping is used to restrict movement when the character is in the air. not realistic, but feels better

- regular damping is used on the ground - use a small value

- various forces and limits to play around with

- Sphere radius: dynamics sphere radius size

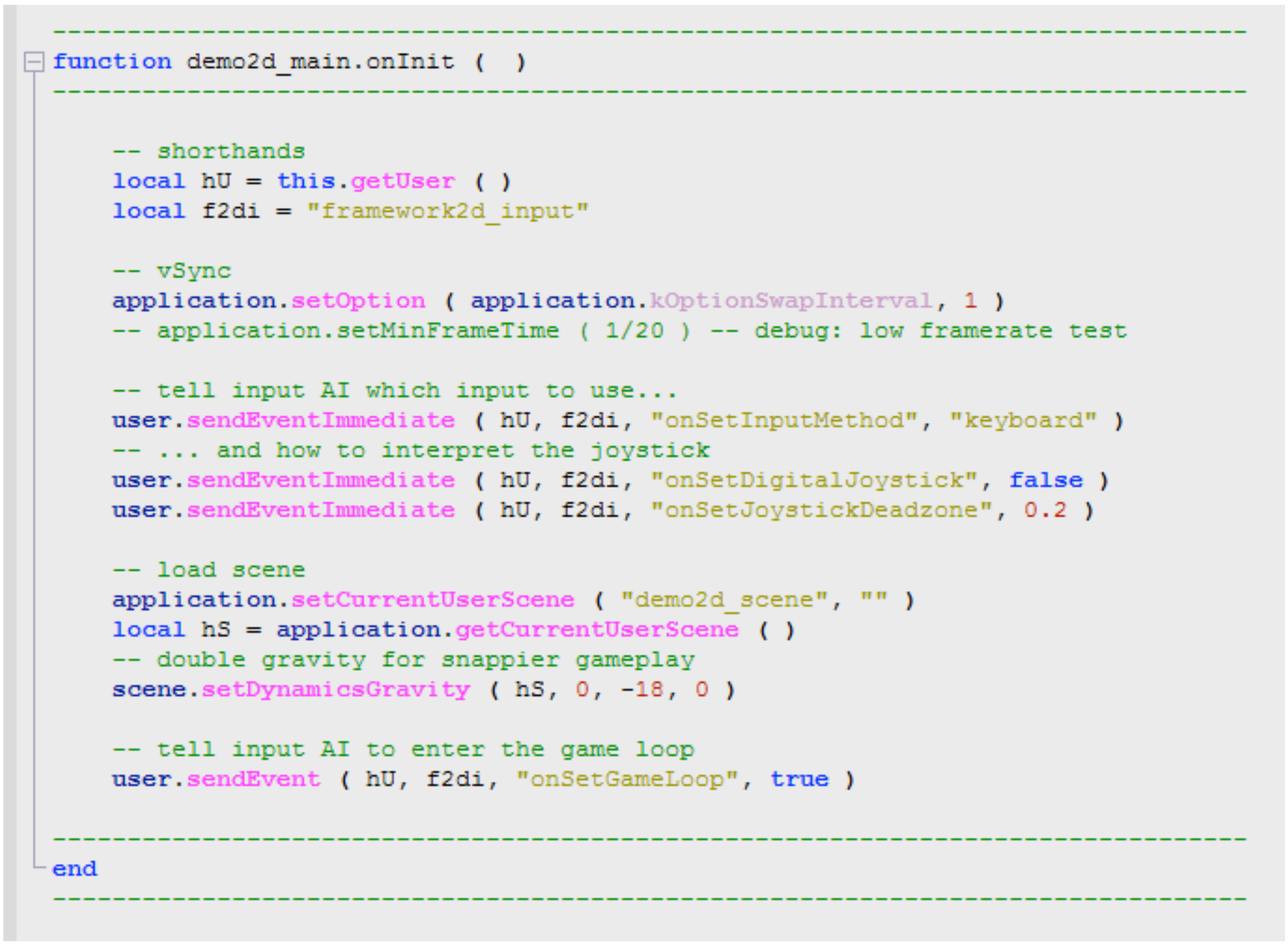
- player model string of a referenced model in the game, will be loaded dynamically

- adjust the guardbox in initDynamics if you must

-all input is evaluated in onEnterFrame, if you don’t want any more input, cut the source in the input AI model, see above

DEMO INIT

- player AI is designed for -18 gravity, not standard gravity
- call the _input handlers to set up input methods
- when everything is set up, set the GAME state
- VSYNC recommended



```lua
--------------------------------------------------------------------------------
function demo2d_main.onInit (  )
--------------------------------------------------------------------------------


    -- shorthands
    local hU = this.getUser ( )
    local f2di = "framework2d_input"

    -- vSync
    application.setOption ( application.kOptionSwapInterval, 1 )
    -- application.setMinFrameTime ( 1/20 ) -- debug: low framerate test

    -- tell input AI which input to use...
    user.sendEventImmediate ( hU, f2di, "onSetInputMethod", "keyboard" )
    -- ... and how to interpret the joystick
    user.sendEventImmediate ( hU, f2di, "onSetDigitalJoystick", false )
    user.sendEventImmediate ( hU, f2di, "onSetJoystickDeadzone", 0.2 )

    -- load scene
    application.setCurrentUserScene ( "demo2d_scene", "" )
    local hS = application.getCurrentUserScene ( )
    -- double gravity for snappier gameplay
    scene.setDynamicsGravity ( hS, 0, -18, 0 )

    -- tell input AI to enter the game loop
    user.sendEvent ( hU, f2di, "onSetGameLoop", true )

--------------------------------------------------------------------------------
end
--------------------------------------------------------------------------------
```