#include "pch.h"

#include "GameController.h"

using namespace DirectX::SimpleMath;

GameController::GameController(void) :

gamepadState ( )

, connected (false)

{

}

GameController::~GameController(void)

{

}

void GameController::Update()

{

ZeroMemory( &this->gamepadState, sizeof(XINPUT\_STATE) );

DWORD dwResult = XInputGetState(0, &this->gamepadState);

if (dwResult == ERROR\_SUCCESS)

{

this->connected = true;

}

else

{

this->connected = false;

}

}

bool GameController::IsConnected()

{

return this->connected;

}

bool GameController::IsButtonDown(unsigned short int button)

{

if ( this->gamepadState.Gamepad.wButtons & button )

{

return true;

}

else

{

return false;

}

}

bool GameController::IsTriggerDown(unsigned short int trigger)

{

if ( this->gamepadState.Gamepad.bLeftTrigger & trigger )

{

return true;

}

else

{

return false;

}

}

Vector2 GameController::GetThumbstick(unsigned short int thumbstick)

{

float thumbstickX = 0;

float thumbstickY = 0;

float deadzone = 0;

const float maximum = 32767.0f;

//Create a vector pointing in same direction as thumbstick; initialize to zero

// vector as we'll return the zero vector if the thumbstick is in the deadzone.

Vector2 thumbstickVector = Vector2(0.0f);

if (XINPUT\_GAMEPAD\_LEFT\_THUMB & thumbstick)

{

thumbstickX = this->gamepadState.Gamepad.sThumbLX;

thumbstickY = this->gamepadState.Gamepad.sThumbLY;

deadzone = (float)XINPUT\_GAMEPAD\_LEFT\_THUMB\_DEADZONE;

}

else if (XINPUT\_GAMEPAD\_RIGHT\_THUMB & thumbstick)

{

thumbstickX = this->gamepadState.Gamepad.sThumbRX;

thumbstickY = this->gamepadState.Gamepad.sThumbRY;

deadzone = (float)XINPUT\_GAMEPAD\_RIGHT\_THUMB\_DEADZONE;

}

//Calculate haw far the thumbstick is displaced from its rest position

float magnitude = sqrt(thumbstickX \* thumbstickX + thumbstickY \* thumbstickY);

if ( magnitude > deadzone )

{

//Make thumbstickVector a vector pointing in same direction as the thumbstick

thumbstickVector.x = thumbstickX;

thumbstickVector.y = -thumbstickY;

//Use the SimpleMath::Vector2::Normalize method to scale our vector to a unit vector

thumbstickVector.Normalize();

//Finally, scale this in proportion to the original magnitude, ignoring the deadzone

thumbstickVector = thumbstickVector \* ( (magnitude - deadzone) / (maximum - deadzone) );

}

return thumbstickVector;

}

void GameController::Vibrate(int leftVal, int rightVal)

{

// Create a Vibraton State

XINPUT\_VIBRATION Vibration;

// Zeroise the Vibration

ZeroMemory(&Vibration, sizeof(XINPUT\_VIBRATION));

// Set the Vibration Values

Vibration.wLeftMotorSpeed = leftVal;

Vibration.wRightMotorSpeed = rightVal;

// Vibrate the controller

XInputSetState(\_controllerNum, &Vibration);

}

short GameController::GetTrigger()

{

short leftTrigger = 0;

Vector2 convertNumber = Vector2(0.0f);

leftTrigger = this->gamepadState.Gamepad.bLeftTrigger;

return leftTrigger;

}