UCSO API Walkthrough

### In this walkthrough, we’ll add UCSO to the default ShuttlePB. You can find the completed example on [GitHub](https://github.com/abdullah-radwan/ShuttlePB_UCSO). Note that the completed example is upgraded to VESSEL4.

### First, include UCSO, then make an instance of it. Add clbkConsumeBufferedKey to intercept keyboard commands, and clbkDrawHUD to draw on the HUD. Dots represent another code, which isn’t related to UCSO. You need to add these in their correct location.

#include "UCSO\_API.h"

………………………

public:

………………………

int clbkConsumeBufferedKey(DWORD key, bool down, char\* kstate);

bool clbkDrawHUD(int mode, const HUDPAINTSPEC\* hps, oapi::Sketchpad\* skp);

private:

………………………

UCSO\* ucso;

int cargoIndex; // For cargo selection on the HUD

};

### Initiate the UCSO instance in the constructor, and delete it in the destructor.

ShuttlePB::ShuttlePB(OBJHANDLE hVessel, int flightmodel) : VESSEL3(hVessel, flightmodel)

{

cargoIndex = 0;

ucso = UCSO::CreateInstance(this);

}

ShuttlePB::~ShuttlePB()

{

delete ucso;

}

### ‘

### Add the attachment point and slot to UCSO in clbkSetClassCaps method.

void ShuttlePB::clbkSetClassCaps (FILEHANDLE cfg)

{

………………………

// Set slot number 0 attachment for UCSO

ucso->SetSlotAttachment(0, CreateAttachment(false, { 0,-2.3,-1.3 }, { 0,-1,0 }, { 0,0,1 }, "UCSO"));

}

### Set the keyboard shortcuts for UCSO commands in clbkConsumeBufferedKey.

int ShuttlePB::clbkConsumeBufferedKey(DWORD key, bool down, char\* kstate)

{

if (!down) return 0; // If the key is let go (not pressed)

if (KEYMOD\_SHIFT(kstate)) { // If Shift key is pressed

switch (key)

{

case OAPI\_KEY\_A:

ucso->AddCargo(cargoIndex);

return 1;

case OAPI\_KEY\_G:

ucso->GrappleCargo();

return 1;

case OAPI\_KEY\_R:

ucso->ReleaseCargo();

return 1;

case OAPI\_KEY\_U:

ucso->UnpackCargo();

return 1;

case OAPI\_KEY\_S:

// Drain the required mass to fill the tank, by subtracting the maximum mass from the current mass

SetFuelMass(GetFuelMass() + ucso->UseResource("Fuel", GetMaxFuelMass() - GetFuelMass()));

return 1;

case OAPI\_KEY\_D:

ucso->DeleteCargo(0);

return 1;

default:

break;

}

}

if (key == OAPI\_KEY\_S) {

// Reset the index if reached the cargo count, otherwise increase the index

cargoIndex + 1 < ucso->GetAvailableCargoCount() ? cargoIndex++ : cargoIndex = 0;

return 1;

}

return 0;

}

### Draw the keyboard shortcuts on the HUD, the selected cargo, and the grappled cargo information.

bool ShuttlePB::clbkDrawHUD(int mode, const HUDPAINTSPEC\* hps, oapi::Sketchpad\* skp)

{

// Draw the default HUD (Surface, Orbit, etc...)

VESSEL4::clbkDrawHUD(mode, hps, skp);

// Determine the screen ratio

int s = hps->H;

double d = s \* 0.00130208;

int sw = hps->W;

int lw = static\_cast<int>(16 \* sw / 1024);

int x = 0;

if (s / sw < 0.7284) x = (lw \* 10) + 10;

int y = static\_cast<int>((168 \* d) + (-88 \* d));

// Set the color to green

skp->SetTextColor(0x0066FF66);

char buffer[256];

sprintf(buffer, "Selected cargo: %s", ucso->GetAvailableCargoName(cargoIndex));

skp->Text(x, y, buffer, strlen(buffer));

y += 32;

skp->Text(x, y, "S = Select cargo to add", 23);

y += 16;

skp->Text(x, y, "Shift + A = Add the selected cargo", 34);

y += 16;

skp->Text(x, y, "Shift + G = Grapple the nearest cargo", 37);

y += 16;

skp->Text(x, y, "Shift + R = Release the grappled cargo", 38);

y += 16;

skp->Text(x, y, "Shift + U = Unpack the nearest cargo", 36);

y += 16;

skp->Text(x, y, "Shift + S = Use the grappled fuel cargo", 39);

y += 16;

skp->Text(x, y, "Shift + D = Delete the grappled cargo", 37);

// End here if no vessel is attached

if (GetAttachmentHandle(true, 0)) if(!GetAttachmentStatus(GetAttachmentHandle(true, 0))) return true;

struct UCSO::CargoInfo cargoInfo = ucso->GetCargoInfo(0);

if (!cargoInfo.valid) return true;

y += 32;

skp->Text(x, y, "Grappled cargo information", 26);

y += 32;

sprintf(buffer, "Mass: %f", cargoInfo.mass);

skp->Text(x, y, buffer, strlen(buffer));

switch (cargoInfo.type)

{

case UCSO::STATIC:

skp->Text(x, y - 16, "Type: Static", 12);

break;

case UCSO::RESOURCE:

skp->Text(x, y - 16, "Type: Resource", 14);

y += 16;

sprintf(buffer, "Resource type: %s", cargoInfo.resourceType);

skp->Text(x, y, buffer, strlen(buffer));

y += 16;

sprintf(buffer, "Resource mass: %f", cargoInfo.resourceMass);

skp->Text(x, y, buffer, strlen(buffer));

y += 16;

break;

case UCSO::UNPACKABLE:

skp->Text(x, y - 16, "Type: Unpackable", 16);

y += 16;

sprintf(buffer, "Spawn module: %s", cargoInfo.spawnModule);

skp->Text(x, y, buffer, strlen(buffer));

y += 16;

switch (cargoInfo.unpackMode)

{

case UCSO::LANDED:

skp->Text(x, y, "Unpack mode: Landed", 19);

break;

case UCSO::DELAYED:

skp->Text(x, y, "Unpack mode: Delayed", 20);

y += 16;

sprintf(buffer, "Unpack delay: %is", cargoInfo.unpackDelay);

skp->Text(x, y, buffer, strlen(buffer));

break;

case UCSO::MANUAL:

skp->Text(x, y, "Unpack mode: Manual", 19);

break;

}

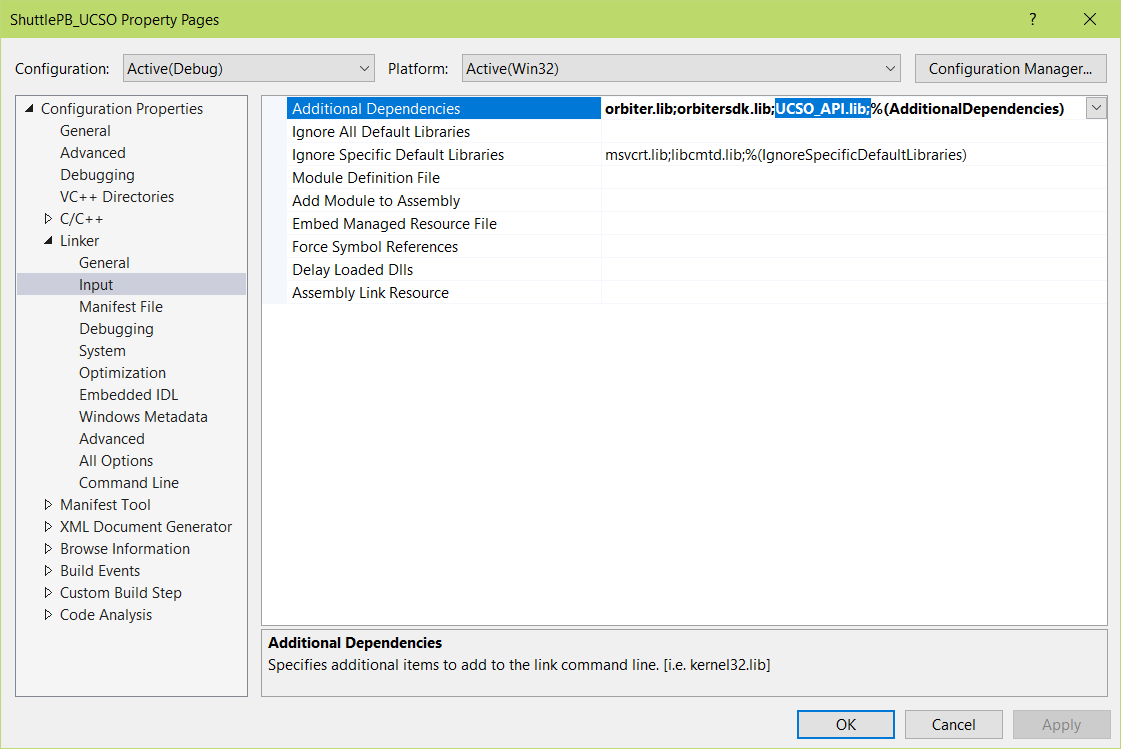
break;

}

return true;

}

### You need now to add the UCSO library to the linker. Open the project settings by right-clicking on the bold ShuttlePB on the right, and choosing ‘Properties’. Then go to Linker -> Input. Add ‘UCSO\_API.lib;’ to ‘Additional Dependencies’. Do this for both configurations (Release and Debug).



### Compile the project now, it should compile without errors. Open any scenario and spawn ShuttlePB and some cargoes. Have fun!