UCSO Custom Cargo API Walkthrough

###### In this walkthrough, we will create a UCSO custom cargo from scratch. The cargo is a lamp that emits light when unpacked. The cargo mesh and texture is ready to use, and the Visual Studio project is set as detailed in the manual. You can find the complete example in ‘Orbitersdk\samples\UCSO\LampCargo’ folder. The complete example has more features.

Start by opening the vessel header file, including the custom cargo API, and inheriting UCSO::CustomCargo class:

#include <UCSO\_CustomCargo.h>

class LampCargo : public VESSEL4, public UCSO::CustomCargo

You must override the pure virtual methods and methods that belong to the cargo (e.g. DrainResource for a resource cargo, etc.). You can override other methods such as the CargoGrappled method if you need it.

class LampCargo : public VESSEL4, public UCSO::CustomCargo

{

public:

LampCargo(OBJHANDLE hVessel, int flightmodel);

void clbkSetClassCaps(FILEHANDLE cfg) override;

void clbkLoadStateEx(FILEHANDLE scn, void\* status) override;

void clbkPreStep(double simt, double simdt, double mjd) override;

void clbkSaveState(FILEHANDLE scn) override;

CargoInfo GetCargoInfo() override;

OBJHANDLE GetCargoHandle() override;

ATTACHMENTHANDLE GetCargoAttachmentHandle() override;

bool UnpackCargo() override;

bool UnpackCargo(bool once);

private:

// This struct is used to hold the beacon information, as AddBeacon method takes a pointer.

// The pointer must be valid during the vessel life. If the variable is local, it will be invalid as soon as the method ends.

// A class variable will be valid for the vessel life.

// The beacon is added to have the 'lamp light' effect, but spotlight is actually lighting.

struct

{

VECTOR3 pos = { 0, 2.55, 0.65 };

VECTOR3 color = { 1,1,1 };

BEACONLIGHTSPEC beaconSpec = { BEACONSHAPE\_DIFFUSE, &pos, &color, 2, 0.2, 0, 0, 0, false };

} beaconStruct;

T

// It is not necessary to use a struct for the spotlight, but to keep things organized.

// This struct is static because AddSpotLight method will make a copy of it, so it can be the same for all instances.

// In the beacon struct, AddBeacon method takes a pointer so every instance must have its own data to enable and disable the beacon.

static const struct SpotStruct

{

const VECTOR3 pos = { 0, 2.55, 0.5 };

// Tilt the light 20 degrees down

const double tilt = -20 \* RAD;

const VECTOR3 dir = { 0, sin(tilt), cos(tilt) };

const double range = 10;

const double att0 = 0.001;

const double att1 = 0;

const double att2 = 0.005;

const double umbra = 45 \* RAD;

const double penumbra = PI05;

const COLOUR4 diffuse = { 1,1,1,0 };

const COLOUR4 specular = { 1,1,1,0 };

const COLOUR4 ambient = { 0,0,0,0 };

} spotStruct;

static const double unpackedSize;

SpotLight\* spotLight = nullptr;

UCSO::CustomCargo::CargoInfo cargoInfo;

ATTACHMENTHANDLE attachmentHandle = nullptr;

void SetPackedCaps();

void SetUnpackedCaps(bool init = true);

};

In the source file, set the DLL callbacks for Orbiter and set the class variables.

#include "LampCargo.h"

#include <sstream>

const double LampCargo::unpackedSize = 4;

const LampCargo::SpotStruct LampCargo::spotStruct;

DLLCLBK VESSEL\* ovcInit(OBJHANDLE hvessel, int flightmodel) { return new LampCargo(hvessel, flightmodel); }

DLLCLBK void ovcExit(VESSEL\* vessel) { if (vessel) delete static\_cast<LampCargo\*>(vessel); }

LampCargo::LampCargo(OBJHANDLE hVessel, int flightmodel) : VESSEL4(hVessel, flightmodel)

{

// Set cargo information

cargoInfo.type = UCSO::CustomCargo::UNPACKABLE\_ONLY;

cargoInfo.unpacked = false;

cargoInfo.breathable = false;

cargoInfo.spawnCount = 5;

cargoInfo.unpackedHeight = 2.9;

}

Set the cargo capabilities in clbkSetClassCaps method.

void LampCargo::clbkSetClassCaps(FILEHANDLE cfg)

{

SetEnableFocus(false);

AddBeacon(&beaconStruct.beaconSpec);

spotLight = static\_cast<SpotLight\*>(AddSpotLight(spotStruct.pos, spotStruct.dir, spotStruct.range,

spotStruct.att0, spotStruct.att1, spotStruct.att2, spotStruct.umbra, spotStruct.penumbra,

spotStruct.diffuse, spotStruct.specular, spotStruct.ambient));

// Set the cargo properties.

// It is set here (not in clbkLoadStateEx) because that method won't be called if the vessel is spawned in the simulator.

// So if it is spawned in the simulator, set its default properties which is a packed cargo.

SetPackedCaps();

}

Load the cargo status if it is being loaded from a scenario.

void LampCargo::clbkLoadStateEx(FILEHANDLE scn, void\* status)

{

char\* line;

while (oapiReadScenario\_nextline(scn, line))

{

std::istringstream ss;

ss.str(line);

std::string data;

if (ss >> data)

{

if (data == "Unpacked") ss >> cargoInfo.unpacked;

else ParseScenarioLineEx(line, status);

if (cargoInfo.unpacked) SetUnpackedCaps(false);

}

else ParseScenarioLineEx(line, status);

}

}

Save the cargo status to the scenario.

void LampCargo::clbkSaveState(FILEHANDLE scn)

{

VESSEL4::clbkSaveState(scn);

oapiWriteScenario\_int(scn, "Unpacked", cargoInfo.unpacked);

}

Set the custom cargo methods.

UCSO::CustomCargo::CargoInfo LampCargo::GetCargoInfo() { return cargoInfo; }

OBJHANDLE LampCargo::GetCargoHandle() { return GetHandle(); }

ATTACHMENTHANDLE LampCargo::GetCargoAttachmentHandle() { return attachmentHandle; }

bool LampCargo::PackCargo() { cargoInfo.unpacked = false; SetPackedCaps(); return true; }

bool LampCargo::UnpackCargo() { cargoInfo.unpacked = true; SetUnpackedCaps(); return true; }

Set the packed cargo capabilities.

void LampCargo::SetPackedCaps()

{

// Don't proceed if unpacked

if (cargoInfo.unpacked) return;

// Disable the beacon and spotlight

beaconStruct.beaconSpec.active = false;

spotLight->Activate(false);

// Replace the unpacked mesh with the packed mesh

InsertMesh("UCSO//Container3", 0);

SetEmptyMass(185);

SetSize(0.65);

ClearAttachments();

// Create the cargo packed attachment

attachmentHandle = CreateAttachment(true, { 0, -0.65, 0 }, { 0, -1, 0 }, { 0, 0, 1 }, "UCSO");

SetPMI({ 0.28, 0.28, 0.28 });

SetCrossSections({ 1.69, 1.69, 1.69 });

double stiffness = GetMass() \* G \* 1000;

double damping = 0.9 \* (2 \* sqrt(GetMass() \* stiffness));

// Values are pre-set for 1.3m size and 0.65 height

TOUCHDOWNVTX tdvtx[4] =

{

{{ 1.3, -0.65, -0.012 }, stiffness, damping, 3, 3},

{{ 0, -0.65, 0.65 }, stiffness, damping, 3, 3},

{{ -1.3, -0.65, -0.012 }, stiffness, damping, 3, 3},

{{ 0, 19.5, 0 }, stiffness, damping, 3, 3}

};

SetTouchdownPoints(tdvtx, 4);

}

Set the unpacked cargo capabilities.

void LampCargo::SetUnpackedCaps()

{

// Enable the beacon and spotlight

beaconStruct.beaconSpec.active = true;

spotLight->Activate(true);

InsertMesh("UCSO//Lamp", 0);

SetSize(unpackedSize);

SetEmptyMass(50);

ClearAttachments();

attachmentHandle = CreateAttachment(true, { 0, -cargoInfo.unpackedHeight, -0.2 }, { 0, 1, 0 }, { 0, 0, 1 }, "UCSO");

SetPMI({ 1.44, 0.2, 1.56 });

SetCrossSections({ 0.58, 0.24, 0.6 });

double stiffness = GetMass() \* G \* 1000;

double damping = 0.9 \* (2 \* sqrt(GetMass() \* stiffness));

double sizeSin = -sin(30 \* RAD) \* unpackedSize;

double sizeCos = cos(30 \* RAD) \* unpackedSize;

TOUCHDOWNVTX tdvtx[4] =

{

{ { sizeCos, -cargoInfo.unpackedHeight, sizeSin}, stiffness, damping, 3, 3},

{ { 0, -cargoInfo.unpackedHeight, unpackedSize }, stiffness, damping, 3, 3},

{ { -sizeCos, -cargoInfo.unpackedHeight, sizeSin }, stiffness, damping, 3, 3},

{ { 0, 15 \* unpackedSize, 0 }, stiffness, damping, 3, 3}

};

SetTouchdownPoints(tdvtx, 4);

}

Compile the project, set the mesh, texture, and configuration file as detailed in the manual, and spawn your cargo in a scenario. Have fun!