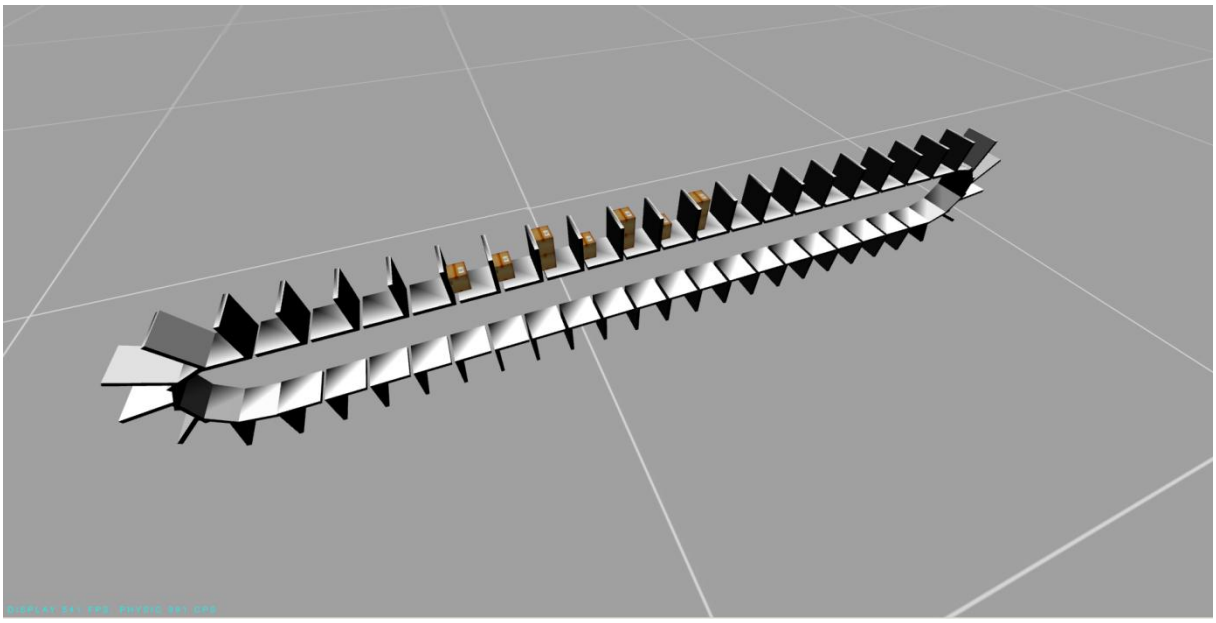
Chain simulation with Virtual Universe Pro

The purpose is to simulate chained objects. The solution explained in this technical note is to use a small C program to move the objects. The objects of the chain could be created into Virtual Universe Pro. The duplicate objects "as a chain" permits to create easily these items. The objects positions and rotations must be right defined, the C program will use these positions and rotations to interpolate the positions of the chain items during the simulation. If the chain objects are imported from Solidworks, be sure to select "Convert Solidworks matrix to objects positions and rotations" in the import properties into Virtual Universe Pro. The chain objects can be used into the physics engine. The "use physic" and also "force position to physic engine" properties must be set to true.

These example requires Virtual Universe Pro 2.013 or newer version.

Example of a chain simulation :



This example is present in the Technical tips / chain Virtual Universe Pro examples directory.

The C program is embedded into a C code behavior. It is written into a .C text file which is present in the media folder.

Regarding the structure, a 3D sprite (without drawing) must be parent of the C code behavior, a behavior used to store a parameter called "cycletime" and all the 3D sprite used as chain items. The cycletime value defines the global time in seconds for the chain for making one cycle.

The C program works as follow:
- at the beginning of the simulation ("startup" function), objects belonging to the same parent than the C code behavior are browsed and each objects ids, positions and rotations are stored in an array.

- when running, according to a "cycletime" value stored in a behavior and the elapsed time, the objects are move and rotated.

```
// Source of the C code program

#include "windows.h"

int vu_getvalsprite3d(unsigned b,char *name,double *result);
int vu_setvalsprite3d(unsigned b,char *name,double v);
int vu_getfirstsprite3d(unsigned b,char *name);
int vu_getnextsprite3d(unsigned b,char *name,int after);
int vu_getbehavior(unsigned b,char *name,double *result);


// Variables
double advance=0; // From 0 to 1 : a complete cycle

#define MAXCHAINLENGTH 4096

int chainitemsid[MAXCHAINLENGTH];

double chainitemsx[MAXCHAINLENGTH];
double chainitemsy[MAXCHAINLENGTH];
double chainitemsz[MAXCHAINLENGTH];
double chainitemsrx[MAXCHAINLENGTH];
double chainitemsry[MAXCHAINLENGTH];
double chainitemsrz[MAXCHAINLENGTH];

unsigned nchainitems=0;


DWORD starttime;

void startup(unsigned b)
{
int s;
unsigned count;
nchainitems=0;
// enum chain items
s=vu_getfirstsprite3d(b,".\\");
while(s>=0)
      {
      chainitemsid[nchainitems++]=s;
      s=vu_getnextsprite3d(b,".\\",s);
      }
// get chain items positions and rotations
for(count=0;count<nchainitems;count++)
      {
      char str[16];
      sprintf(str,"[#%u].POSX",chainitemsid[count]);
      vu_getvalsprite3d(b,str,&chainitemsx[count]);
      sprintf(str,"[#%u].POSY",chainitemsid[count]);
      vu_getvalsprite3d(b,str,&chainitemsy[count]);
      sprintf(str,"[#%u].POSZ",chainitemsid[count]);
      vu_getvalsprite3d(b,str,&chainitemsz[count]);
      sprintf(str,"[#%u].ROTX",chainitemsid[count]);
      vu_getvalsprite3d(b,str,&chainitemsrx[count]);
      sprintf(str,"[#%u].ROTY",chainitemsid[count]);
      vu_getvalsprite3d(b,str,&chainitemsry[count]);
      sprintf(str,"[#%u].ROTZ",chainitemsid[count]);
      vu_getvalsprite3d(b,str,&chainitemsrz[count]);
      }
starttime=GetTickCount();
}
```

```
// Interpolate between 2 angles
double interpolateangle(double coef,double a1,double a2)
{
double a=a1-a2;
if(a>180)
      {
      a=360-a;
      a1=a+a2;
      }
else
      {
      if(a<-180)
            {
            a=-360+a;
            a1=a+a2;
            }
      }
return (a1*(1-coef))+(a2*coef);
}


void loop(unsigned b)
{
double cycletime=0; // Time to make one complete cycle (each chain items
will pass to the start positions) in seconds, stored in the behavior
"cycletime" must be >0
vu_getbehavior(b,"..\\cycletime",&cycletime);
if(cycletime<=0) return;
unsigned count;
// Calculate elapsed time in ms
DWORD dwellapsed=GetTickCount()-starttime;
double ellapsed;
if(dwellapsed==0)
      {
      return; // Nothing to do
      }
// Calculate elapsed time in seconds
ellapsed=dwellapsed;
ellapsed/=1000;
// Elapsed must be between 0 and cycletime
while(ellapsed-cycletime>=0) ellapsed-=cycletime;
// Advance is between 0 and 1
advance=ellapsed/cycletime;
double step;
// Calculate step time between to items
step=1;
step/=nchainitems;
for(count=0;count<nchainitems;count++)
      {
      // Calculate a coef for intepoltaing between these 2 items
      unsigned upos=(unsigned)(nchainitems*advance);
      double pos=upos;
      pos*=step;
      double coef=(advance-pos)/step;
      double localadvance=count;
      localadvance*=step;
      localadvance+=advance;
      if(localadvance>1) localadvance-=1;
```

```
        // The 2 items positions between the position to calculate
        unsigned i1,i2;
        i1=(unsigned)(nchainitems*localadvance);

        char debugstr[128];
        sprintf(debugstr,"%u %f %f %d",upos,localadvance,advance,i1);
        vu_out(b,debugstr);

        i2=i1+1;
        if(i2==nchainitems) i2=0;
        if(coef<0) coef=0;
        if(coef>=1) coef=0.9999;
        double posx=chainitemsx[i1]*(1-coef)+chainitemsx[i2]*coef;
        double posy=chainitemsy[i1]*(1-coef)+chainitemsy[i2]*coef;
        double posz=chainitemsz[i1]*(1-coef)+chainitemsz[i2]*coef;
        double rotx=interpolateangle(coef,chainitemsrx[i1],chainitemsrx[i2]);
        double roty=interpolateangle(coef,chainitemsry[i1],chainitemsry[i2]);
        double rotz=interpolateangle(coef,chainitemsrz[i1],chainitemsrz[i2]);
        char str[16];
        sprintf(str,"[#%u].POSX",chainitemsid[count]);
        vu_setvalsprite3d(b,str,posx);
        sprintf(str,"[#%u].POSY",chainitemsid[count]);
        vu_setvalsprite3d(b,str,posy);
        sprintf(str,"[#%u].POSZ",chainitemsid[count]);
        vu_setvalsprite3d(b,str,posz);
        sprintf(str,"[#%u].ROTX",chainitemsid[count]);
        vu_setvalsprite3d(b,str,rotx);
        sprintf(str,"[#%u].ROTY",chainitemsid[count]);
        vu_setvalsprite3d(b,str,roty);
        sprintf(str,"[#%u].ROTZ",chainitemsid[count]);
        vu_setvalsprite3d(b,str,rotz);
        }
}

void clean(unsigned b)
{
}
```