voxcraft

Release v2020

Sida Liu

Jan 06, 2021

CONTENTS

CHAPTER

ONE

OVERVIEW

This is an overview of the project.

CHAPTER

TWO

INSTALL

2.1 Files needed to run

2.1.1 Voxelyze3

Voxelyze3 is an executable that split tasks into multiple groups, and call **vx_node_worker** to execute them on different GPUs. **vx_node_worker** works with **Voxelyze3** and is the one who actually carry out the calculation. (So don't delete it.)

2.1.2 VoxCAD

VoxCAD can visualize **history** file recorded via **Voxelyze3**. **VoxCAD** should be run locally, and GPU is only optional for **VoxCAD**.

2.2 Install Voxelyze3

2.2.1 On DeepGreen

DeepGreen is UVM's GPU cluster. We have already compiled on DeepGreen, so it will be quite easy to use **Voxelyze3** on DeepGreen.

Follow a five-minute instruction here: https://github.com/liusida/gpuVoxels-dg-installation

2.2.2 On Desktop/Laptop with GPUs

If you have access to root, the best way to get started is using Docker, since Nvidia provide ready to use CUDA docker image files.

First, install Docker CE (Community Edition).

Then, install nvidia-container:

```
sudo apt-get update
sudo apt-get install -y nvidia-container-runtime nvidia-container-toolkit
```

Then, start the container:

```
docker run --gpus all --name=gpuVoxels --volume=/tmp:/gpuVoxels/host -it sidaliu/
↪gpuvoxels:1.0
```

Note: You can copy **Voxelyze3** and **vx_node_worker** to wherever you want and run it there.

Note: You can change the path /tmp to anywhere you want, and keep your code and data there.

Note: If you are not familiar with Docker, here's some helpful commands you will use.

```
# The commands below should be run on host command line
# See what containers are running
docker ps
# See what containers are running/suspending
docker ps --all
# You should see a suspended container called gpuVoxels
# Let's re-enter that container
docker start gpuVoxels
docker attach gpuVoxels
```

2.2.3 How I made the docker image

If you'd like to see what's inside the image **sidaliu/gpuvoxels:1.0**, here is the way I build it.

```
docker run --gpus all --name=gpuVoxels --volume=/tmp:/host -it nvidia/cuda:10.1-devel-
↪ubuntu18.04
```

The meaning of last command is run a docker container using nvidia image cuda, download it if not exists. The name of container is gpuVoxels, and /tmp of the host is shared in the container as /host.

After a while...

Now, I am inside the docker container.

```
apt update
apt install -y git cmake libboost-all-dev
cd /
mkdir gpuVoxels
git clone https://github.com/liusida/gpuVoxels.git gpuVoxels_src
cd /gpuVoxels_src/Voxelyze3
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release -DCUDA_DEBUG=OFF ..
cmake --build .
cp Voxelyze3 /gpuVoxels
cp vx_node_worker /gpuVoxels
exit
# Back to the host
```

```
docker commit -a "Sida Liu <sliu1@uvm.edu>" gpuVoxels sidaliu/gpuvoxels:1.0
docker push sidaliu/gpuvoxels:1.0
```

2.2.4 On Server with GPUs but without root

It will be too tricky to install all the dependencies without root, especially CUDA 10.x.

If there's already CUDA 10.1 or 10.2 on the server, you'll need to compile and install cmake, g++, boost from source.

2.3 Install VoxCAD

VoxCAD need OpenGL, Boost, QT5. You will need to build VoxCAD from source. Here are instructions for Ubuntu and Mac.

2.3.1 On Ubuntu

The package names in Ubuntu are:

OpenGL (libglfw3-dev, freeglut3-dev, libglm-dev, mesa-utils), Boost (libboost-all-dev), QT5 (qtbase5-dev).

Here I demostrate how to do it in Ubuntu 18.04.

```
sudo apt install -y git cmake libboost-all-dev qtbase5-dev libglfw3-dev freeglut3-dev␣
↪libglm-dev mesa-utils
# this will take a while...
git clone https://github.com/liusida/gpuVoxels.git gpuVoxels_src
cd /gpuVoxels_src/VoxCAD
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
cmake --build .
# done.
```

2.3.2 On Mac

```
brew install cmake
brew install boost
brew install qt5
brew install glfw3
brew cask install xquartz
brew install freeglut
brew install glm
brew install mesa

# this will take a while...

git clone https://github.com/liusida/gpuVoxels.git gpuVoxels_src
cd gpuVoxels_src/VoxCAD/
mkdir build
cd build/
cmake -DCMAKE_BUILD_TYPE=Release ..
```

Wait a minute, you need to make a little bit of hacking in the source files:

first is *VoxCAD/src/QTUtils/QOpenGL.cpp*

```
Change line 11 from
    #include "GL/glu.h"
To
    #include "OpenGL/glu.h"
```

second is *VoxCAD/CMakeFiles.txt*

```
Change line 70 from
    find_package(glm CONFIG REQUIRED) # glm
to
    # find_package(glm CONFIG REQUIRED) # glm

Change line 44 from
    target_link_libraries(VoxCAD PRIVATE ${OpenGL_LIBRARIES} GL)
to
    target_link_libraries(VoxCAD PRIVATE ${OpenGL_LIBRARIES})
```

OK, let's continue.

```
cmake --build .
# done.
```

Thank Arlo Cohen at University of Vermont for providing the instruction on Mac.

2.3.3 Running VoxCAD

Run VoxCAD using command:

```
./VoxCAD <filename.history>
```

Note: Now you can copy the file **VoxCAD** and **Default.vxc** to wherever you want and run it there.

Note: If you forget the file **Default.vxc**, the simulation will stuck at every step.

CHAPTER

THREE

TRY

Google Colab provides a free online GPU environment.

```
# First, go to menu->Runtime->Change runtime type, select GPU.
# Then, run the script:
!git clone https://github.com/liusida/gpuVoxels.git; cd gpuVoxels/; git checkout dev;␣
↪git pull;
!mkdir build; cd build; rm * -rf; cmake ../gpuVoxels/Voxelyze3; make -j 10;
!cd build; ./Voxelyze3 -i ../gpuVoxels/zoo/basic/ -o output.xml -f > ../a.history
print("Done! Start downloading the generated a.history. Please use your local VoxCAD␣
↪to open the file.")
from google.colab import files
files.download('a.history')
```

Readers can also check this readonly example:

https://colab.research.google.com/drive/1yiqw7Uq3W3CgYCinXq4t808M2l7uuLv1?usp=sharing

CHAPTER

FOUR

RUN

./Voxelyze3 -i data/ -o output.xml -lf

CHAPTER

FIVE

OVERVIEW

VXA/VXD is the format we use to define and customize our simulations.

base.vxa is the base for a generation of simulations.

xxxx.vxd is the customized configuration for a single simulation.

CHAPTER

SIX

VXA TAGS

Note: Here is a sample base.vxa file.

6.1 VXA.GPU

Configuration related to GPU settings.

6.1.1 VXA.GPU.HeapSize

The proportion of memories in GPUs that we want to use as Heap Memory. If we use the GPU both for display and computing, like on my laptop, we can choose *0.5*; if we use the GPU sololy for computing, like on a GPU server, we can use *1.0*.

CUDA manages all memories in GPUs. The default heap size is 8M bytes, which is too small if we want to use *malloc* in CUDA code.

See CUDA Documentation

6.2 VXA.Simulator

Configuration related to the whole simulation settings.

6.2.1 VXA.Simulator.FitnessFunction

Value: a MathTree (refer to MathTree)

The fitness function for a creature. The simulator will sort all creatures in one generation by their fitness scores and report them as an output file. (refer to Voxelyze3 parameter -o)

6.2.2 VXA.Simulator.Integration.DtFrac

Value: 0.0 ~ 1.0

Simulator will compute a recommended time step for each simulation, but sometimes the simulation still exploded even using the recommended time step.

So we can define a fraction here to say that we want to use even smaller time step to make sure the simulation runs safely.

1.0 if everything is fine, *0.5* if we want a safer run.

6.2.3 VXA.Simulator.Damping.BondDampingZ

Value: 0.0 ~ 1.0

This is a name used in VX1. In VX2 it's called *InternalDamping* or *zetaInternal*.

When we compute the force for the link, we multiply the calculated force by a *dampingMultiplier()*, which is related to this internal damping variable.

In another words, if we want the voxels to jiggle a lot, we use *0.0*, if we want the voxels to calm down, we use *1.0*.

6.2.4 VXA.Simulator.Damping.ColDampingZ

Value: 0.0 ~ 1.0

This is a name used in VX1. In VX2 it's called *CollisionDamping* or *zetaCollision*.

0.0 if we want the voxel bouncing on the floor, *1.0* if we don't want it bouncing.

Note: Why the robot still bounce even when this variable is set to zero?

Because the links between vertical voxels will produce a bouncing force. If there's only one voxel, the bounce will be determined by this variable alone.

6.2.5 VXA.Simulator.Damping.SlowDampingZ

Value: 0.0 ~ 1.0

This is a name used in VX1. In VX2 it's called *GlobalDamping* or *zeta*.

0.0 is no global damping, *1.0* to damp everything in the simulation.

> **Warning:** This is a strong damping term. The whole simulation will almost freeze if this variable is set to *0.1*.

6.2.6 VXA.Simulator.StopCondition.StopConditionFormula

Value: a MathTree (refer to MathTree)

Tell the simulation when to stop. We usually stop at t-4>0, which means stop at 4.0 sec.

```
<mtSUB>
    <mtVAR>t</mtVAR>
    <mtCONST>4</mtCONST>
</mtSUB>
```

6.2.7 VXA.Simulator.RecordHistory.RecordStepSize

Value: an integer

0 if we don't want record any *.history* file. (*.history* file can be played in VoxCAD)

For example, *100* if we want to record the history every 100 steps.

Warning: Recording history will cost much longer time! Now it is about 40x.

6.2.8 VXA.Simulator.RecordHistory.RecordVoxel

Value: 0 or 1

0 if we don't want to record voxels, *1* if we do.

6.2.9 VXA.Simulator.RecordHistory.RecordLink

Value: 0 or 1

0 if we don't want to record links, *1* if we do.

Recording links is usually for debugging porposes. By setting the alpha value of the materials less than 1, we can see the links in the playback.

6.2.10 VXA.Simulator.AttachDetach.EnableCollision

Value: 0 or 1

This variable is controling the voxel-voxel collision. Not related to the collsion with floor.

0 if we don't want voxel-voxel collision, *1* if we do.

Note: Collision detection takes *O(n^2)* time, so disable this feature can make simulation much faster.

6.2.11 VXA.Simulator.AttachDetach.EnableAttach

When collision happens, we can enable attachment. Under certain condition (defined in *AttachCondition*), two voxels will stick together when collide.

6.2.12 VXA.Simulator.AttachDetach.AttachCondition

Value: a set of MathTrees (refer to MathTree)

If we want attachment happens whenever collision happens, we can define *Condition_0*, *Condition_1*, up to *Condition_4*.

```
<Condition_0>
    <mtCONST>1</mtCONST>
</Condition_0>
```

6.2.13 VXA.Simulator.AttachDetach.SafetyGuard

Value: an integer

when attachment happens, there will be a new link formed between two voxels. Sometimes the relative speed of two voxels is too large, the attachment will seem to be unrealistic.

This variable defines the number of steps in which there will be a special damping between two newly attached voxels.

Note: This is the number of steps, not in seconds, so it will change if step size changes.

6.2.14 VXA.Simulator.ForceField

Value: three MathTrees for x,y,z dimension (refer to MathTree)

If we want to apply an external force field to the simulation, we can define it here. We can define *x_forcefield*, *y_forcefield*, and *z_forcefield*.

Here is an example to define a force field that only has value on x direction.

x_forcefield = *0-x*, which means everything will be pull to y axis.

```
<x_forcefield>
    <mtSUB>
        <mtCONST>0</mtCONST>
        <mtCONST>x</mtCONST>
    </mtSUB>
</x_forcefield>
<y_forcefield>
    <mtCONST>0</mtCONST>
</y_forcefield>
```

6.2.15 VXA.Simulator.EnableSignals

Value: 0 or 1

0 if we want to disable singals. *1* if we want to enable the singals.

6.2.16 VXA.Simulator.EnableCilia

Value: 0 or 1

0 if we want to disable cilia. *1* if we want to enable the cilia.

6.2.17 VXA.Simulator.SavePositionOfAllVoxels

Value: 0 or 1

0 if we don't want the output report XML file contains the final positions of all voxels, *1* if we do.

6.2.18 VXA.Simulator.MaxDistInVoxelLengthsToCountAsPair

Value: a real number with no unit

Sometimes we need to count how many pairs of TARGET voxels are close to each other. By defining this variable, we can specify the threshold for counting.

0 if we don't want to count close pairs.

Note: This quantity is the distance over average voxel length. For example, if the voxel length is 0.01 meter, then if we set this variable to 2 here, it means the distance is 2*0.01 meter.

Value: 0 or 1

0 if we don't want to enable counting closeness

6.3 VXA.Environment

Configuration related to the whole virtual environment.

6.3.1 VXA.Environment.Gravity.GravEnabled

Value: 0 or 1

0 if we don't want gravity, *1* if we want gravity.

Note: If we disable the gravity here, we can still use the Force Field to define a downward force that is identical to gravity.

However, if we enable force field will be a little bit slower than simply using this gravity variable. (refer to force field)

6.3.2 VXA.Environment.Gravity.GravAcc

Value: a real number in m/s^2

-9.81 if we want to use the common gravity on earth. Negative means downward.

6.3.3 VXA.Environment.Gravity.FloorEnabled

Value: 0 or 1

1 if we want there to be a floor at *z=0*, so that thing won't fall forever.

Note: This variable is not related to whether to draw the floor in VoxCAD. That can be controled via a checkbox in VoxCAD.

6.3.4 VXA.Environment.Thermal.VaryTempEnabled

Value: 0 or 1

1 if we want enable temperature.

Note: Why temperature?

One way to make the voxel actuate and let the robot move is to use a varying temperature. In this thermal expansion model, the rest length of the links between voxels will vary due to the temperature change. We can define different coefficient of thermal expansion (CTE) for different materials. (refer to CTE)

6.3.5 VXA.Environment.Thermal.TempAmplitude

Value: a real number in degree Celsius

The amplitude of the temperature oscillation. If we want the voxels to actuate more, we can use larger amplitude.

6.3.6 VXA.Environment.Thermal.TempPeriod

Value: Real number in second

The period of the temperature oscillation. If we want the actuate period to be longer, we can use larger period here.

Note: The temperature will oscillate as a sine wave.

6.4 VXA.VXC

Configuration related to the creatures in the simulation.

6.4.1 VXA.VXC.Lattice.Lattice_Dim

Value: a real number in meter

The dimension (side length) of a voxel.

6.4.2 VXA.VXC.Palette

Palette means the materials.

6.4.3 VXA.VXC.Palette.Material

One material.

6.4.4 VXA.VXC.Palette.Material.Name

Value: a string

The name of the material.

6.4.5 VXA.VXC.Palette.Material.Display

The color of the material.

6.4.6 VXA.VXC.Palette.Material.Display.Red

Value: 0.0 ~ 1.0

6.4.7 VXA.VXC.Palette.Material.Display.Green

Value: 0.0 ~ 1.0

6.4.8 VXA.VXC.Palette.Material.Display.Blue

Value: 0.0 ~ 1.0

6.4.9 VXA.VXC.Palette.Material.Display.Alpha

Value: 0.0 ~ 1.0

See RGBA color model

6.4.10 VXA.VXC.Palette.Material.Mechanical.isTarget

Value: 0 or 1

1 if we want voxels made of this material to be the target, *0* if we don't.

Target voxels will trigger special functionalities. For example, when a non-target voxel hit a target voxel, the former one will generate a signal; or, when we use *MeasureFitnessOfTargetMaterialOnly* tag, the fitness function will only take into account the target voxels instead of all voxels.

6.4.11 VXA.VXC.Palette.Material.Mechanical.isMeasured

Value: 0 or 1 (default 1)

1 if we want to measure voxels made by this material in all MathTree functions, especially the fitness function. *0* if we want to exclude this material.

6.4.12 VXA.VXC.Palette.Material.Mechanical.Fixed

Value: 0 or 1

1 if we don't want this material to move at all.

The fixed voxels can serve as the environment, such as wall, steps, etc., or serve as a pin when we want to pin the robot down in space.

6.4.13 VXA.VXC.Palette.Material.Mechanical.sticky

Value: 0 or 1

1 if we want attachment can happen to this material, *0* if we don't.

6.4.14 VXA.VXC.Palette.Material.Mechanical.Cilia

Value: a real number

1 if we want to enable cilia force for this material, *0* if we don't.

Other real number if you want to change the magnitude of the cilia force.

6.4.15 VXA.VXC.Palette.Material.Mechanical.isPaceMaker

Value: 0 or 1

1 if this material can generate periodic signals spontaneously, *0* if not.

6.4.16 VXA.VXC.Palette.Material.Mechanical.PaceMakerPeriod

Value: a real number in second

The pace maker can generate periodic signals spontaneously. The period between two signals is defined by this variable.

6.4.17 VXA.VXC.Palette.Material.Mechanical.signalValueDecay

Value: 0.0 ~ 1.0

When the singal propagates to other part of the body, it has a decay ratio. This variable defines the ratio.

0.0 means the signal cannot propagate at all, *1.0* means the signal never decay and can propagate to infinity.

6.4.18 VXA.VXC.Palette.Material.Mechanical.signalTimeDelay

Value: a real number in second

When the singal propagates to other part of the body, it has a travel speed. The signal may be delayed at every stop (in every voxel). This variable defines how much time it will delay in each voxel.

6.4.19 VXA.VXC.Palette.Material.Mechanical.inactivePeriod

Value: a real number in second

Inspired by the process of action potential in living cells, in which the cell will enter an inactive state for a while to prevent the signal traveling backward. This variable defines the time period that a voxel stays inactive after sending out the signal.

See Action Potential

6.4.20 VXA.VXC.Palette.Material.Mechanical.MatModel

Value: 0 or 1

0 for simple linear elastic model. The mechanical model for a perfectly elastic material is a simple spring. *1* for linear elastic model that can fail. When materials are subjected to a large enough strain they fail by fracture. The voxels will detach. (refer to AttachDetach)

Note: Fail by fracture model need *VXA.Simulator.AttachDetach.EnableDetach* to be 1.

6.4.21 VXA.VXC.Palette.Material.Mechanical.Elastic_Mod

Value: a real number in Pascal

The elastic modulus (a.k.a. Young's Modulus) describes the stiffness of a material. For soft robotics, we usually use 10^7 Pa like a rubber.

See values for common materials

6.4.22 VXA.VXC.Palette.Material.Mechanical.Fail_Stress

Value: a real number in Pascal

When the stress is larger than this threshold, the material fail by fracture.

Note: This need *MatModel* to be *1* and *EnableDetach* to be 1.

6.4.23 VXA.VXC.Palette.Material.Mechanical.Density

Value: a real number in kg/m^3

For example, natural rubbber's density is about 1.5e+3 kg/m^3.

6.4.24 VXA.VXC.Palette.Material.Mechanical.Poissons_Ratio

Value: 0.0 ~ 0.5

The ratio of the proportional decrease in a lateral measurement to the proportional increase in length in a sample of material that is elastically stretched.

For example, rubber has a ratio near 0.5, and cork is famous for a ratio near 0.0.

6.4.25 VXA.VXC.Palette.Material.Mechanical.CTE

Value: a small real number in 1/degree Celsius (same as 1/K)

For example, plastics has CTE of about 10^(-4). To make the actuation more obvious, we can choose CTE to be 0.01. (in reality, we don't have such high CTE material.)

6.4.26 VXA.VXC.Palette.Material.Mechanical.uStatic

Value: 0.0 ~ 5.0

This is a name in VX1. In VX2 it's called *StaticFriction*.

This is the static frictional coefficient. For example, rubber-rubber static friction coefficient is 1.16, and ice-ice static friction coefficient is 0.1.

6.4.27 VXA.VXC.Palette.Material.Mechanical.uDynamic

Value: 0.0 ~ 1.0

This is a name in VX1. In VX2 it's called *KineticFriction*.

This is the kinetic frictional coefficient. For example, rubber-pavement kinetic friction coefficient is 0.8, and steel-ice kinetic friction coefficient is 0.01.

6.4.28 VXA.VXC.Palette.Material.Mechanical.Cilia

Value: 0 or 1

0 if this material does not exerting cilia force, *1* if it has.

6.4.29 VXA.VXC.Structure

The simulation start with a world of voxels, which are in a lattice structure.

We should always use the attribute *Compression="ASCII_READABLE"*, since by doing that we can see the data directly.

6.4.30 VXA.VXC.Structure.X_Voxels

Value: an integer larger than zero

The x dimension of the voxel world.

6.4.31 VXA.VXC.Structure.Y_Voxels

Value: an integer larger than zero

The y dimension of the voxel world.

6.4.32 VXA.VXC.Structure.Z_Voxels

Value: an integer larger than zero

The z dimension of the voxel world.

6.4.33 VXA.VXC.Structure.Data

This section defines material type for each position in the lattice.

The x and y dimension combined together form a layer. The first layer is z=1, the second layer is z=2, aranged from bottom up.

6.4.34 VXA.VXC.Structure.Data.Layer

Value: a string of integers, which define the material for each voxel in one layer.

The number is the index of material but in charactor type, *0* for nothing. For example, the first material corresponds to '1', the nine-th material corresponds to '9'.

If we have more than 9 materials, we can continue using ':',';','<'... following '9' which is in the ASCII order.

See ASCII order here

The length of digits in one layer should equal X_Voxels * Y_Voxels.

The number of layers should equal Z_Voxels.

6.4.35 VXA.VXC.Structure.PhaseOffset

One way to make the voxel actuate is using thermal expansion model, with *VaryTempEnabled* and *CTE*, the rest length of voxels will change due to temperature changing.

However, if all voxels actuate in phase, it is quite difficult to find any interesting behaviors.

Phase offset introduce more interesting behavior by allowing each voxel has its own phase.

Phase offset settings should have the same dimension as *Data*, and each value corresponds to the phase offset of that voxel.

6.4.36 VXA.VXC.Structure.PhaseOffset.Layer

Value: a set of real number 0.0 ~ 1.0

6.4.37 VXA.VXC.Structure.BaseCiliaForce

Value: a set of real numbers, which define the cilia force in x,y,z dimension for each voxel in one layer.

Note: This feature works with *EnableCilia* = 1 and only apply to material with *Cilia* = 1.

6.4.38 VXA.VXC.Structure.ShiftCiliaForce

Value: a set of real numbers, which define the behavior shifting of the cilia force in x,y,z dimension for each voxel in one layer.

When a voxel has a signal larger than 0, there will be a shifting in behavior.

6.5 VXA.RawPrint

Value: a string

What was passed here will be simply passed along to the history file (or standard output).

CHAPTER

SEVEN

ZOO

Here are demos of the simulation.

7.1 Basic

7.1.1 Video

If you can't see a YouTube video? click here instead.

7.1.2 How to make this demo?

Put two files into a folder *data*.

base.vxa

```
<VXA Version="1.1">
<Simulator>
    <RecordHistory>
        <RecordStepSize>100</RecordStepSize>
        <RecordVoxel>1</RecordVoxel>
        <RecordLink>0</RecordLink>
    </RecordHistory>
    <StopCondition>
        <StopConditionFormula>
            <mtSUB>
            <mtVAR>t</mtVAR>
            <mtCONST>1</mtCONST>
            </mtSUB>
        </StopConditionFormula>
    </StopCondition>
</Simulator>
<VXC Version="0.94">
    <Lattice>
        <Lattice_Dim>0.01</Lattice_Dim>
    </Lattice>
    <Palette>
        <Material>
            <Name>Default</Name>
            <Display>
```

(continues on next page)

(continued from previous page)

```
                <Red>1</Red>
                <Green>0</Green>
                <Blue>0</Blue>
                <Alpha>1</Alpha>
            </Display>
            <Mechanical>
                <MatModel>0</MatModel>
                <Elastic_Mod>1e+05</Elastic_Mod>
                <Fail_Stress>0</Fail_Stress>
                <Density>1e+3</Density>
                <Poissons_Ratio>0.35</Poissons_Ratio>
                <CTE>0</CTE>
                <MaterialTempPhase>0</MaterialTempPhase>
                <uStatic>1</uStatic>
                <uDynamic>0.8</uDynamic>
            </Mechanical>
        </Material>
    </Palette>
    <Structure Compression="ASCII_READABLE">
        <X_Voxels>1</X_Voxels>
        <Y_Voxels>1</Y_Voxels>
        <Z_Voxels>2</Z_Voxels>
        <Data>
            <Layer><![CDATA[0]]></Layer>
            <Layer><![CDATA[1]]></Layer>
        </Data>
    </Structure>
</VXC>
</VXA>
```

Note: Short Explanation

There are two layers: the 1st one has no element, and the 2nd one has a voxel made of the 1st material, so when the simulation starts, the voxel falls and bounces without any damping.

robot.vxd

```
<VXD>
</VXD>
```

Note: Short Explanation

Here we only need one robot, so nothing to customize.

For the meanings of the tags, please refer to VXA Tags.

Linux Command

After we prepared the *data/* folder, we can run the program to get the visualization.

```
./Voxelyze3 -i data > a.history
./VoxCAD a.history
```

For the meanings of the executables, please refer to Run.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search