

# Three-Dimensional Simulation of Morphogenesis

Jesus A. Izaguirre

Department of Computer Science and  
Engineering

University of Notre Dame

# What is Morphogenesis?

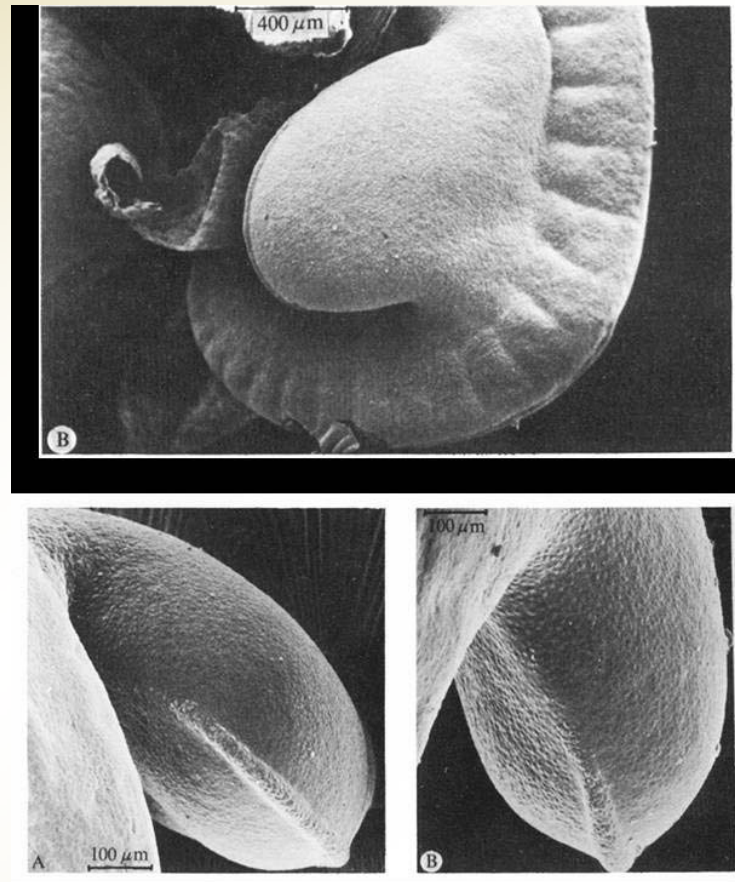
- A stage in embryonic development
- Mesenchymal cells begin to cluster and form patterns. Involves:
  - Cell differentiation
  - Cell growth
  - Cell division
  - Cell migration
  - Chemical secretion/resorption/diffusion

# Basic Cell Sorting Model

- Two different types of cells
- One type is very adhesive to other cells of the same type
- All cells repelled by the medium

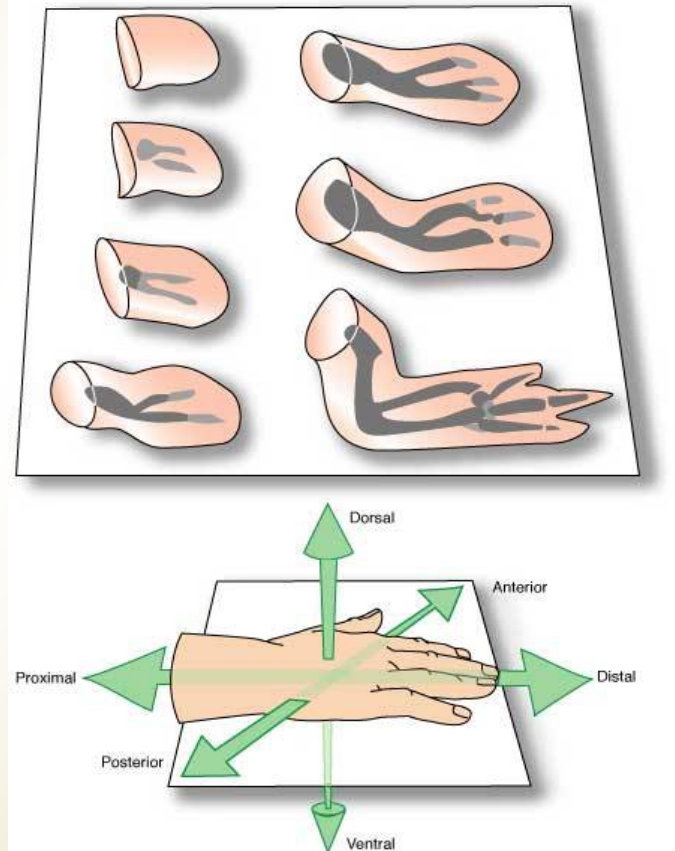
# Example: Avian Limb

Chick limb bud, after 3.5 days

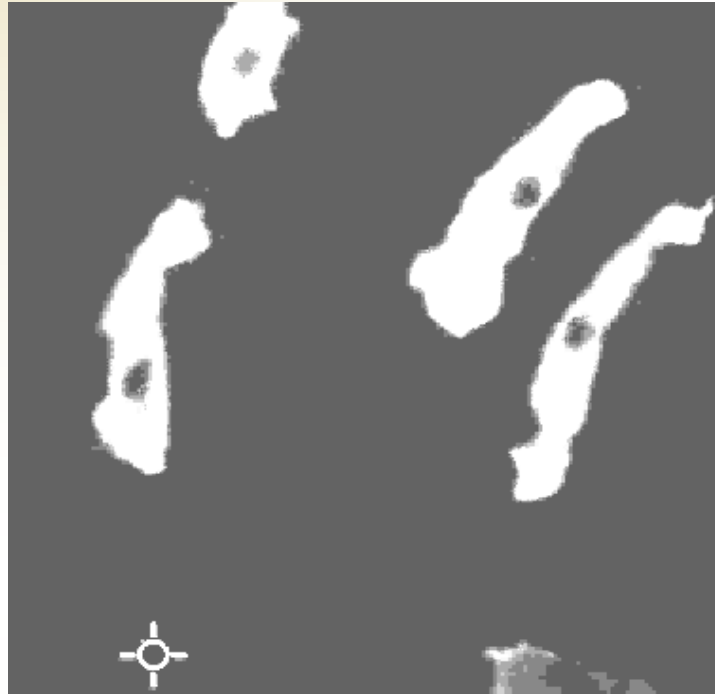


# Avian Limb Stages

- Schematic Representation
- Forelimb Pattern Formation Order:
  - Humerus
  - Radius/Ulna
  - Carpals/Metacarpals
  - Digits



# Example: Dictyostelium Discoideum



# Mathematical Modeling

- First step: find a biologically relevant mathematical model
- One well defined model is the Cellular Potts Model (CPM)

# Cellular Potts Model (CPM)

- Cells represented in a 3D lattice
- Each unique cell given a different integer index, indices stored in pixels
- Extracellular matrix has index of 0
- Neighbors and levels (1-4) are given for a pixel S

				2										
	1	1	2	2	2									
1	1	1	1	2		3								
1	1	1	1		3	3	3							
	1	1		3	3	3	3	3						
					3	3	3							
						3								
E	C	M	=	0				N <sub>4</sub>	N <sub>3</sub>	N <sub>4</sub>				
								N <sub>4</sub>	N <sub>2</sub>	N <sub>1</sub>	N <sub>2</sub>	N <sub>4</sub>		
								N <sub>3</sub>	N <sub>1</sub>	S	N <sub>1</sub>	N <sub>3</sub>		
								N <sub>4</sub>	N <sub>2</sub>	N <sub>1</sub>	N <sub>2</sub>	N <sub>4</sub>		
								N <sub>4</sub>	N <sub>3</sub>	N <sub>4</sub>				

# Metropolis Algorithm of the CPM

- Choose a pixel at random
- Propose to change the pixel's index to that of one of its neighbors (index 'flip')
- Execute the flip with Monte Carlo probability based on the resulting energy from the flip:

$$P(\Delta E) = \begin{cases} 1, & \text{if } \Delta E \leq 0, \\ e^{-\Delta E/kT}, & \text{if } \Delta E > 0, \end{cases}$$

# CPM Energy Calculation

- Three terms:

$$E = E_{Contact} + E_{Volume} + E_{Chemical}$$

- $E_{Contact}$  results from adhesion between adjacent cells
- $E_{Volume}$  results from deviation of cells from their target volume and surface
- $E_{Chemical}$  results from cell chemotaxis or haptotaxis to a secreted or diffusing chemical.

# CPM Energy Equations

$$E_{Contact} = \sum_{(i,j,k),(i',j',k')} J_{\tau(\sigma),\tau'(\sigma')} \cdot (1 - \delta(\sigma(i,j,k), \sigma'(i',j',k')))$$

$$E_{Volume} = \sum_{cells} \lambda_{\sigma} (v(\sigma, \tau) - v_{target}(\sigma, \tau))^2 + \sum_{cells} \lambda'_{\sigma} (s(\sigma, \tau) - s_{target}(\sigma, \tau))^2$$

$$E_{Chemical} = \sum_{x,y,z} \mu(\sigma(x,y,z)) \cdot C(x,y,z)$$

# Key Differences Between Simulations

- Cell Sort
  - Basic CPM adhesion and volume
  - No chemical energy
- Avian Limb
  - Cells undergo *haptotaxis* with chemical fibronectin
  - Domain grows
- Dictyostelium Discoideum
  - Polarity within cells
  - Activator field is dynamic

# Cell Type Maps

- Specify:
  - A set of cell types to which each cell can belong
  - A set of cell state variables that each cell contains
  - A set of rules for a cell to change between types
  - Cell's type determines its behavior

# Cell Type Maps

- Cell Sort
  - Two cell types: Light and Dark (50/50 odds at simulation startup)
  - Dark cells are very adhesive to one another, all cells are very repellent with the medium
- Avian Limb
  - Two cell types: NonCondensing and Condensing
  - Condensing cells are more adhesive
- Dictyostelium Discoideum
  - Three cell types: Prespore, Prestalk and Autocycling
  - Only Autocycling cells react to the activator
  - Each cell type is adhesive with other cells of the same type, Prespore cells cluster to form a spore, prestalk to form a stalk and Autocycling to form the tip

# Activator Chemical Patterns

- Established by ODEs/PDEs
- Turing's continuum reaction diffusion approach

$$\frac{\partial \vec{u}}{\partial t} = D \nabla^2 \vec{u} + F(\vec{u})$$

# Ex: Piecewise Puschino Kinetics

An system of coupled reaction-diffusion equations:

$$\begin{aligned}\frac{\partial e}{\partial t} &= \Delta e - f(e) - g, \\ \frac{\partial g}{\partial t} &= D_g \Delta g + \epsilon(e, g)(ke - g),\end{aligned}$$

e: activator

g: inhibitor

f,  $\epsilon$ : piecewise functions



# Activator Chemical Patterns

- Another example: Hentschel/Glimm equations for the Avian limb simulation

$$\begin{aligned}\frac{\partial c_a}{\partial t} &= \gamma[(J_0 + J_a(c_a)\beta(c_a))R_0 - k_a c_a c_i] + b_a(c_a - c_{as})^3 + (d_{ax} \frac{\partial^2 c_a}{\partial x^2} + d_{ay} \frac{\partial^2 c_a}{\partial y^2} + d_{az} \frac{\partial^2 c_a}{\partial z^2}) \\ \frac{\partial c_i}{\partial t} &= \gamma[J_i(c_a)\beta(c_a)R_0 - k_i c_a c_i] + b_i(c_i - c_{is})^3 + D(d_{ix} \frac{\partial^2 c_i}{\partial x^2} + d_{iy} \frac{\partial^2 c_i}{\partial y^2} + d_{iz} \frac{\partial^2 c_i}{\partial z^2})\end{aligned}$$

where  $c_a$  and  $c_i$  represent the respective concentrations of activator and inhibitor,  $c_{as}$  and  $c_{is}$  are the spatially homogeneous steady states for the activator and inhibitor concentrations, and  $R_0$  denotes the average cell density.

# Computational Modeling: Issues

- Software must be extensible, flexible and easy to use, specifically to allow:
  - Extensible CPM Hamiltonians
  - Cell type automata for various organisms
  - Arbitrary number of superimposed chemical fields
- Large 3D CPM Lattices
  - Speed and memory usage concerns

# Addressing These Issues

- CompuCell3D, a three-dimensional C++ framework for morphogenesis simulation

- and -

- BIOLOGO, a domain specific language for morphogenesis, used to extend CompuCell3D

# CompuCell3D Overview

## Initialization and Rendering Plugins

Plugins have access to the entire lattice and can be used to initialize the cells in the lattice and save the state of the lattice at each MC step.

### UniformInitializer

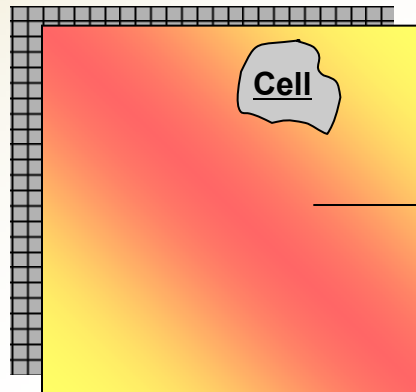
```
<Steppable Type= "UniformInitializer">  
- <Gap>4</Gap>  
- <Width>2</Width>  
</Steppable>
```

### OgleDumper

```
<Steppable Type= "OgleDumper" Frequency= "1">  
- <Renderer>Additive</Renderer>  
- <FilePrefix>chickenlimb</FilePrefix>  
</Steppable>
```

## Lattice and Chemical Fields

The main Potts lattice holds the cells and controls the Monte Carlo loop. Plugins can be used to overlay additional fields, such as chemical concentrations on the lattice.



```
<Potts>  
- <Dimensions x= "71" y= "36" z= "211"/>  
- <Steps>10</Steps>  
- <Temperature>2</Temperature>  
- <Flip2DimRatio>2</Flip2DimRatio>  
</Potts>
```

```
<Plugin Name= "Chemical">  
- <Threshold>0.8</Threshold>  
- <Lambda>10</Lambda>  
- <ConcentrationFile>field.dat  
- </ConcentrationFile></Plugin>
```

## Cell Parameter and Hamiltonian Plugins

Plugins can define new cell parameters and their associated energy functions.

**Volume**  
*volume*  
*volumeEnergy (cell)*

```
<Plugin Name= "Volume">  
- <TargetVolume>64</TargetVolume>  
- <LambdaVolume>0.05</LambdaVolume></Plugin>
```

**Surface**  
*area*  
*surfaceEnergy (cell)*

```
<Plugin Name= "Surface">  
- <TargetSurface>77</TargetSurface>  
- <LambdaSurface>0.05</LambdaSurface></Plugin>
```

**Contact**  
*contactEnergy (*  
*cell11, cell12)*

```
<Plugin Name= "Contact">  
<Energy Type1= "Medium" Type2= "Medium">0</Energy>  
<Energy Type1= "Light" Type2= "Medium">0</Energy>  
<Energy Type1= "Dark" Type2= "Medium">0.1</Energy>  
<Energy Type1= "Light" Type2= "Light">0.5</Energy>  
<Energy Type1= "Dark" Type2= "Dark">3.0</Energy>  
<Energy Type1= "Light" Type2= "Dark">0.5</Energy>  
<Depth>1</Depth></Plugin>
```

# Customizing CompuCell3D

CompuCell3D defines a set of classes that can be extended to add features to a simulation.

**Plugins** are loaded at runtime. They are the main way of adding new features to CompuCell. They can be Steppables, Steppers, CellChangeWatchers, or Automaton.

Some simulation features, such as Renders are so common that they are built into the system.

## CompuCell3D Program Flow

```
Initialization()  
Steppables.Start()  
  
For each Monte Carlo step:  
    For flip attempt:  
        if(flip):  
            CellChangeWatcher(cell)  
            Automaton.Update(cell)  
            Steppers.step()  
        Steppables.step()  
        Renderers.renderStep()  
  
Steppables.finish()
```

**Steppables** are executed once per Monte Carlo step and once before and after the main loop. They are the main hooks for initialization and rendering.

**CellChangeWatchers** are executed once per each successful spin flip. They are useful for adjusting values that depend on the number of lattice points in a cell.

**Automatons** enable cell state to change their state as the simulation evolves.

**Steppers** are executed once per spin flip attempt. They are the main hooks for energy functions.

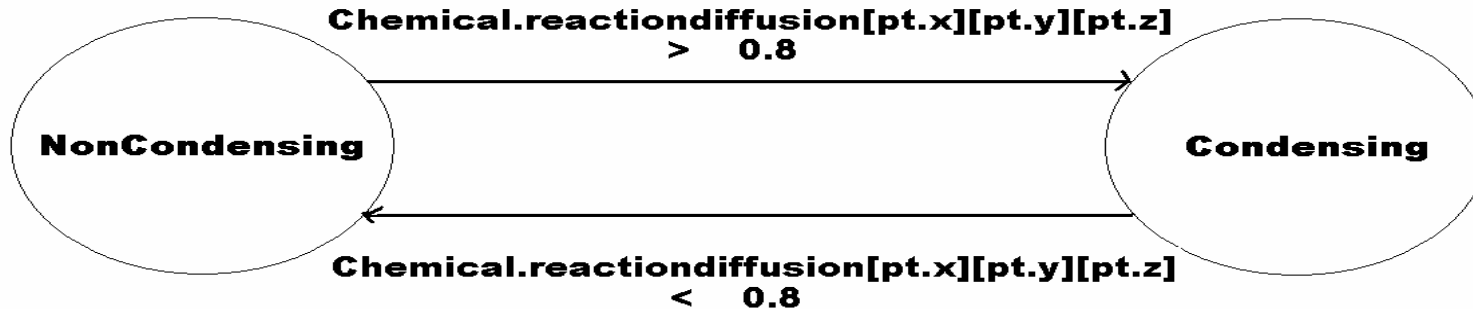
# CompuCell3D Features/Patterns

- Allows different boundary conditions per axis through the Strategy and Factory design patterns
- Dynamic class nodes contiguously allocate all attributes of a particular cell, reducing cache misses and page faults
- Singleton object for medium pixels
- Lazy pixel neighbor evaluation
- Factory pattern for cell object creation

# BIOLOGO

- An XML-based Domain Specific Language
- Language constructs are more understandable to biologists than C++
- After compilation, extensions to CompuCell3D are generated
  - New energy Hamiltonians, automata and fields
  - Only necessary to run BIOLOGO once for the same extensions

## Cell Type Automata

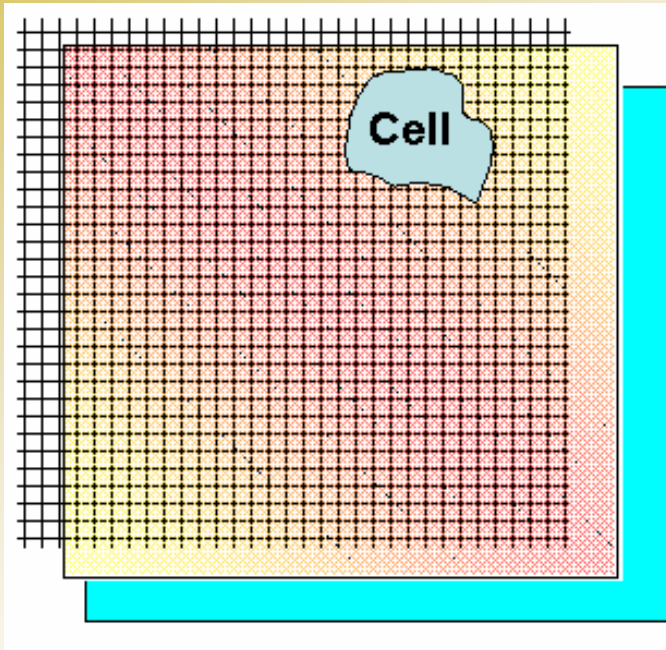


```

<cellmodel name= "Chick">
- <useplugin name= "Chemical" />
- <celltype name= "NonCondensing">
  - <updatecelltypes>
    - <changeif currenttype= "Condensing"
      condition= "Chemical.rd[pt.x][pt.y][pt.z] less 0.8" />
    - </updatecelltypes>
  - </celltype>
- <celltype name= "Condensing">
  - <updatecelltypes>
    - <changeif currenttype= "NonCondensing"
      condition= "Chemical.rd[pt.x][pt.y][pt.z] greater 0.8" />
    - </updatecelltypes>
  - </celltype>
</cellmodel>
    
```

# Representing a Morphogenesis Simulation Through BIOLOGO (cont.)

## Superimposed Chemicals



```

<Hamiltonian name= "ChemicalFibro">
  <Input name= "Threshold" type= "double" />
  <Input name= "Lambda" type= "int" />
  <Input name= "FibroInc" type= "double" />
  <Input name= "ConcentrationFile" type= "file" fieldname= "rd"
    fieldtype= "float" />
  <Field name= "Fibronectin" type= "double" />
  <Step>
    <if condition= "oldcell.type notequal $Medium$" >
      <if condition= "rd[pt.x][pt.y][pt.z] greaterequal Threshold">
        <copy to= "Fibronectin[pt.x][pt.y][pt.z]"
          from= "Fibronectin[pt.x][pt.y][pt.z]+FibroInc" />
      </if>
      <return value= "Fibronectin[pt.x][pt.y][pt.z] * Lambda" />
    </if>
    <return value= "0.0" />
  </Step>
</Hamiltonian>
  
```

CPM Energy Hamiltonians



# Extending CompuCell3D Through BIOLOGO

- Hamiltonians and Automata become CompuCell3D plugins (dynamically loaded)
- Upon extension, these new plugins can be referenced in the CompuCell3D configuration file

# What BIOLOGO generates for CompuCell3D

- Hamiltonians
  - A proxy [Gamma et. al 1995] to register a new plugin
  - Plugin interface (registers an energy function)
  - Step function translated to C++ in a method **changeEnergy()**
  - Accessor methods for all inputs
  - Method to read plugin from configuration file
  - Automake inputs

# What BIOLOGO generates for CompuCell3D

- Automata
  - Dynamic class node for cell state variables
  - A proxy to register the new plugin
  - Plugin interface (registers an automaton)
  - Creation, updatevariables and updatecelltypes modules translated to C++ methods
  - Automake inputs
- Uses a dynamic class node for cell type

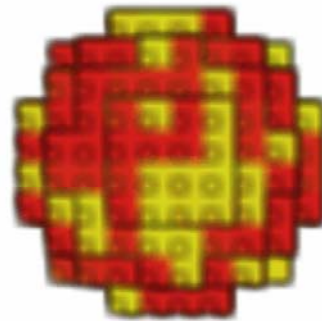
# Avian Limb With Growth



# Dictyostelium Discoideum



# Basic Cell Sort Results



# Current and Future Work

- Currently
  - Irregular geometries
  - Simulating Myxobacteria, which requires cell polarity
  - Scripting capabilities with Python
- Future
  - Integration with chemical equation solvers
  - Better visualization
  - Parallelism

## Acknowledgements

T. Cickovski [4], C. Huang [4], K. Aras [4], J.A. Glazier [1], S.A. Newman[2], M. G. Hentschel[3], M. Alber [6], G. Forgacs[5], B. Kazmierczak [6] , R. Chatuverdi [4], T. Glimm [3]

[1] Departments of Physics and Biology and Biocomplexity Institute, Indiana University, Bloomington

[2] Department of Cell Biology and Anatomy, Basic Science Building, New York Medical College, Valhalla

[3] Department of Physics, Emory University, Atlanta

[4] Department of Computer Science and Engineering, University of Notre Dame, Notre Dame

[5] Department of Physics and Biology, University of Missouri, Columbia

[6] Department of Mathematics, University of Notre Dame, Notre Dame

# More Acknowledgements

- NSF Biocomplexity Grant No. IBN-0083653
- Notre Dame Interdisciplinary Center for the Study of Biocomplexity ([www.nd.edu/~icsb](http://www.nd.edu/~icsb))
- Biocomplexity Institute at Indiana University, Bloomington.

# Appendix: BIOLOGO Files

- cellsort.xml
- chickgrowth.xml
- dicty.xml

# Appendix: Parameters for each simulation

Sim	$\tau_0$	$\tau_1$	$\tau_2$	$J_{\tau_0, \tau_0}$	$J_{\tau_0, \tau_1}$	$J_{\tau_0, \tau_2}$	$J_{\tau_1, \tau_1}$	$J_{\tau_1, \tau_2}$	$J_{\tau_2, \tau_2}$	$J_{T_M, \tau_0}$	$J_{T_M, \tau_1}$	$J_{T_M, \tau_2}$
CS	LT	DK		14	11		2			16	16	
AL	NC	CD		3.0	0.5		0.5			0.1	0.2	
AG	NC	CD		7.0	7.0		0.5			2.9	2.9	
DD	PS	PT	AU	4.5	5.5	6.5	2.5	6	0.5	6.5	5.5	4.5

Sim	$V_T$	$\lambda$	$\mu$	$A_T$	$FR$	T	$L_x$	$L_y$	$L_z$	F/D
CS	20	1				5	50	50	50	1
AL	64	0.05	10	0.8		2	71	36	211	2
AG	16	1	10	0.7	0.01	2	71	31	281	2
DD	4	16	200	0.05		2	50	1	50	64