

Kugel

Mathematics
Formular

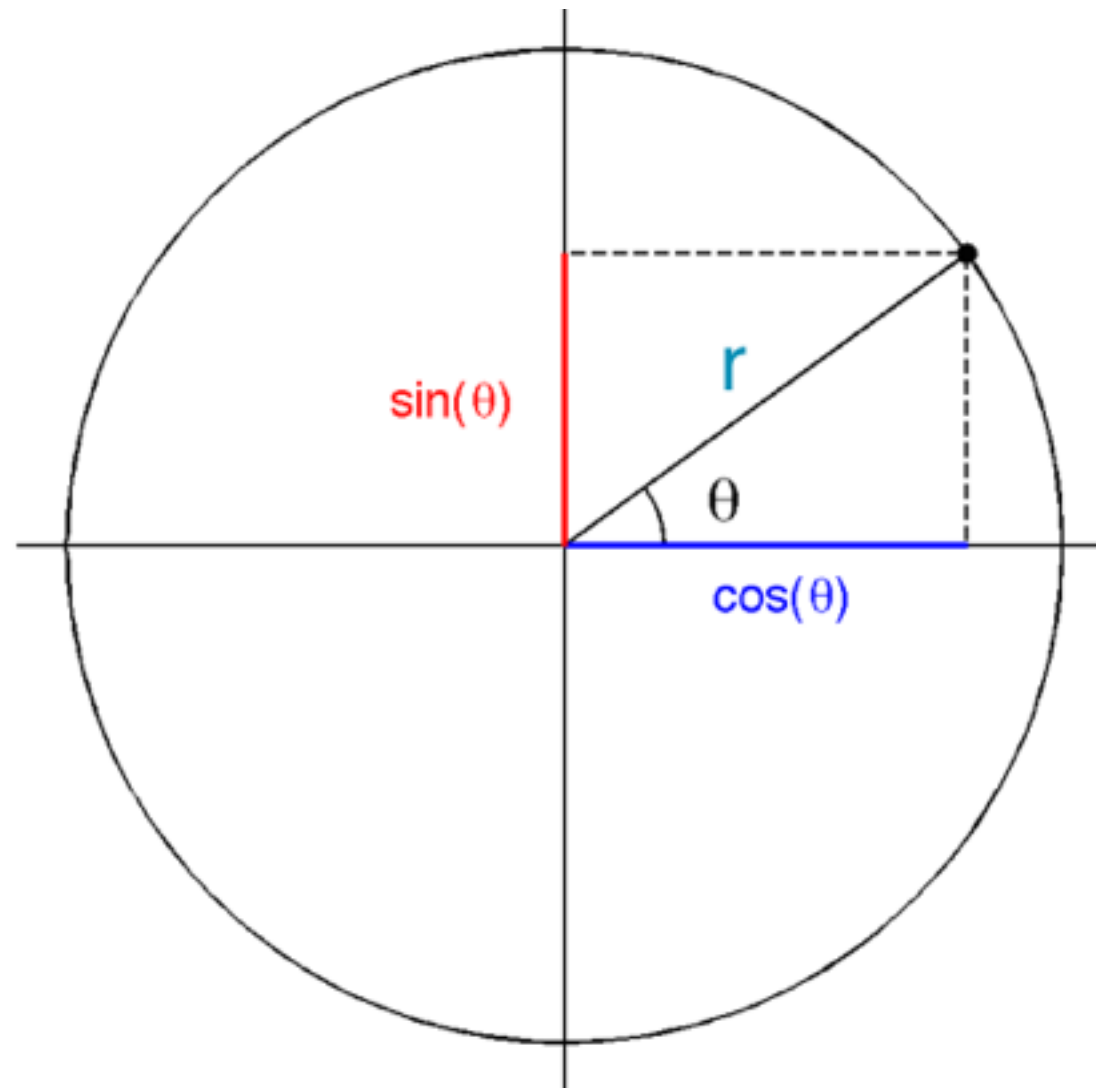


Code

Two Dimensional



Erste Stufe



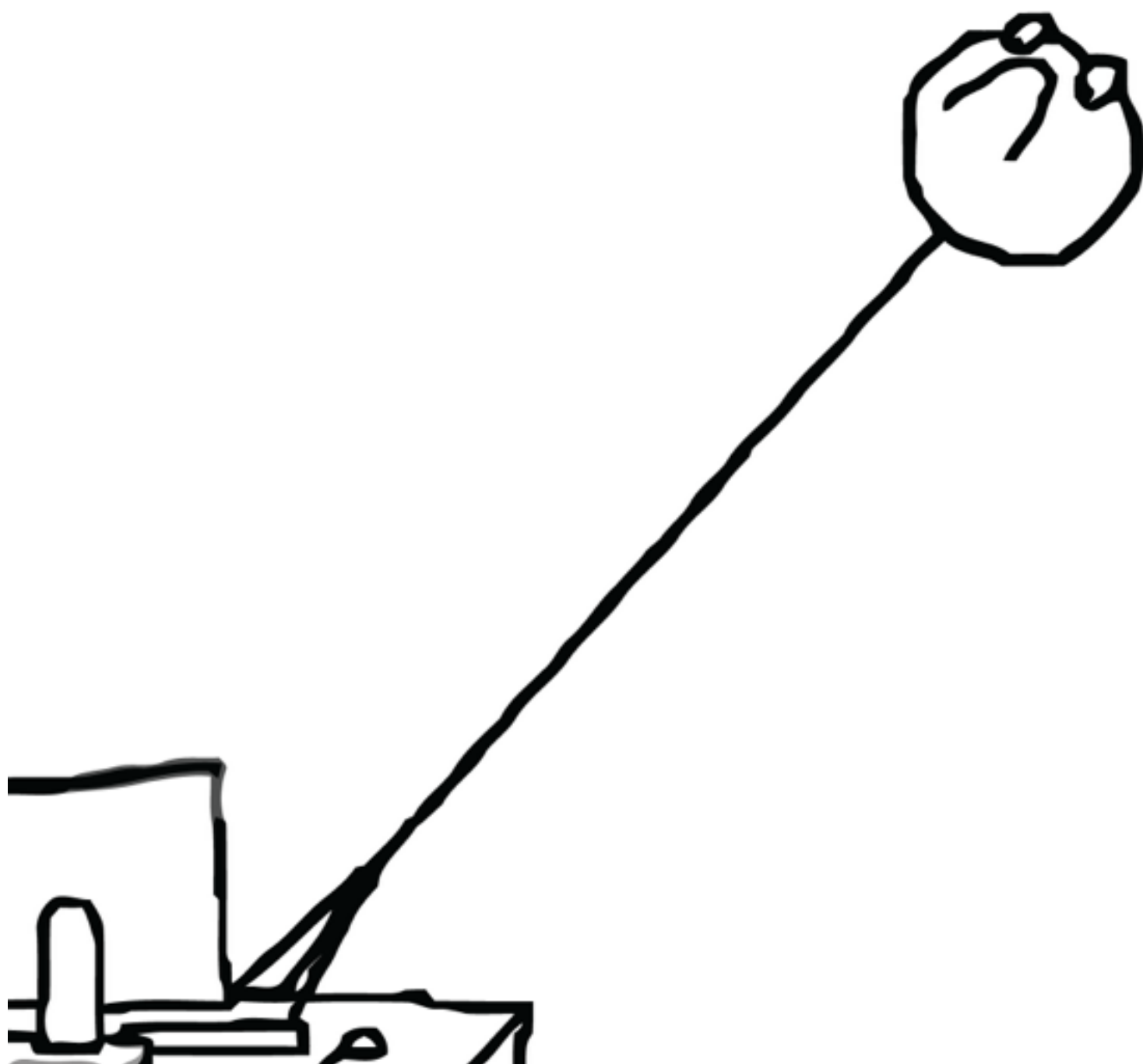
$$x = r * \cos(\theta)$$

$$y = r * \sin(\theta)$$

<http://mathworld.wolfram.com/PolarCoordinates.html>

Cartesian Coordinates

- $x = r * \cos \theta$
- $y = r * \sin \theta$
- θ : degree - ofRandom(θ)
- r : radius



- `vector<ofVec2f> point2D;`
 - `point2dSetting();`
 - `float _theta = ofRandom(0, TWO_PI);`
 - `float _radius = 100;`
 - `float _x = cos(_theta) * _radius;`
 - `float _y = sin(_theta) * _radius;`
 - `ofVec2f _point2D_temp = ofVec2f(_x, _y);`
 - `point2D.push_back(_point2D_temp);`
-
- The diagram illustrates the flow of data and dependencies between variables in the code. A yellow arrow originates from the `point2D` variable in the first line and points to its use in the final `push_back` call. A blue arrow starts from `_theta` in the third line and points to its use in the `cos` function in the fifth line. A green arrow starts from `_radius` in the fourth line and points to its use in the multiplication in the fifth line. Another green arrow starts from `_radius` in the fourth line and points to its use in the multiplication in the sixth line. A red arrow starts from `_x` in the fifth line and points to its use as the first argument in the `ofVec2f` constructor in the seventh line. A second red arrow starts from `_y` in the sixth line and points to its use as the second argument in the `ofVec2f` constructor in the seventh line. A purple arrow starts from `_point2D_temp` in the seventh line and points to its use in the `push_back` call in the final line.

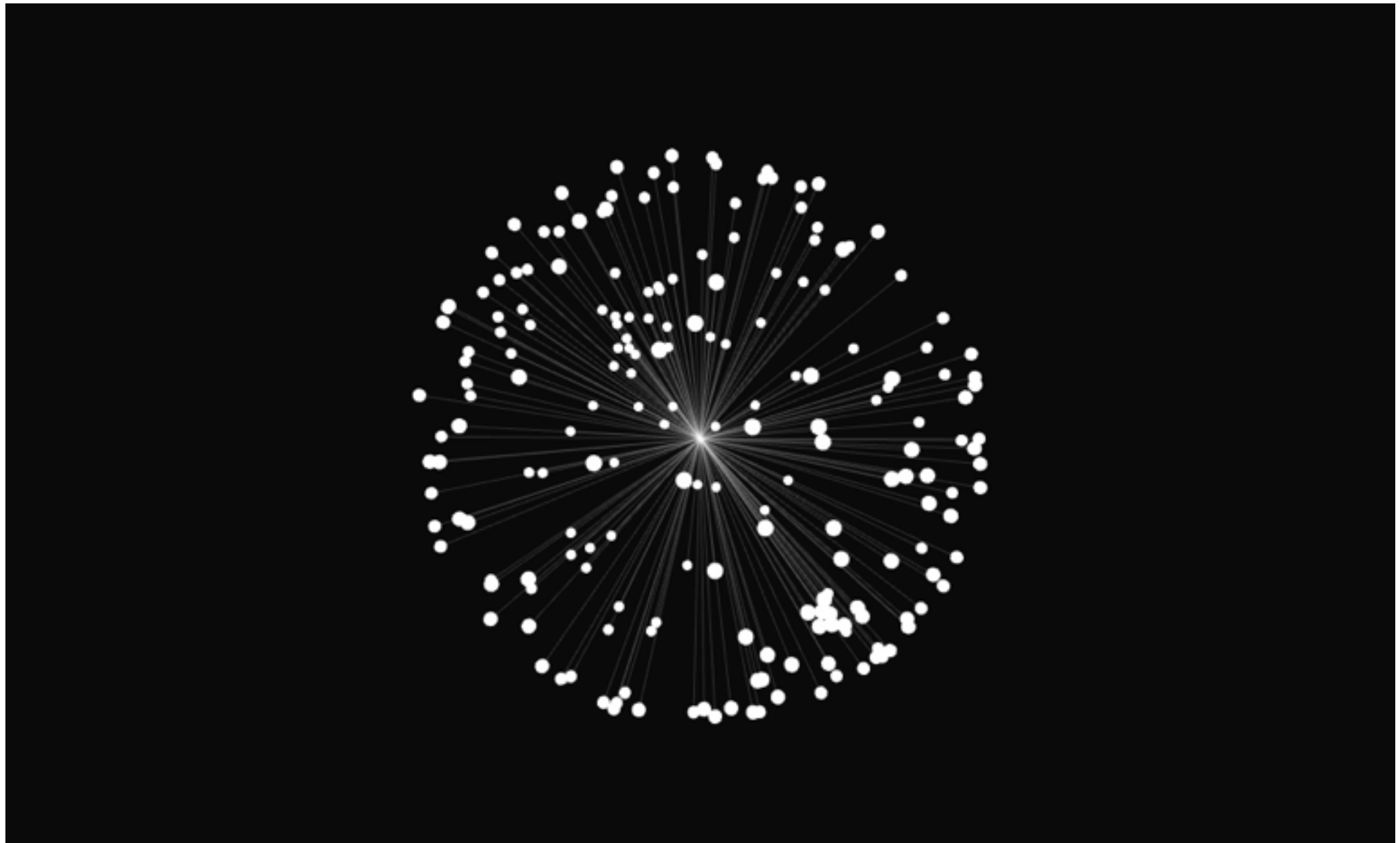
oder

- `float _rad = ofRandom(0, TWO_PI);`
- `float _radius = 100;`
- `point2D.push_back(ofVec2f(cos(_rad) * _radius, sin(_rad) * _radius)));`

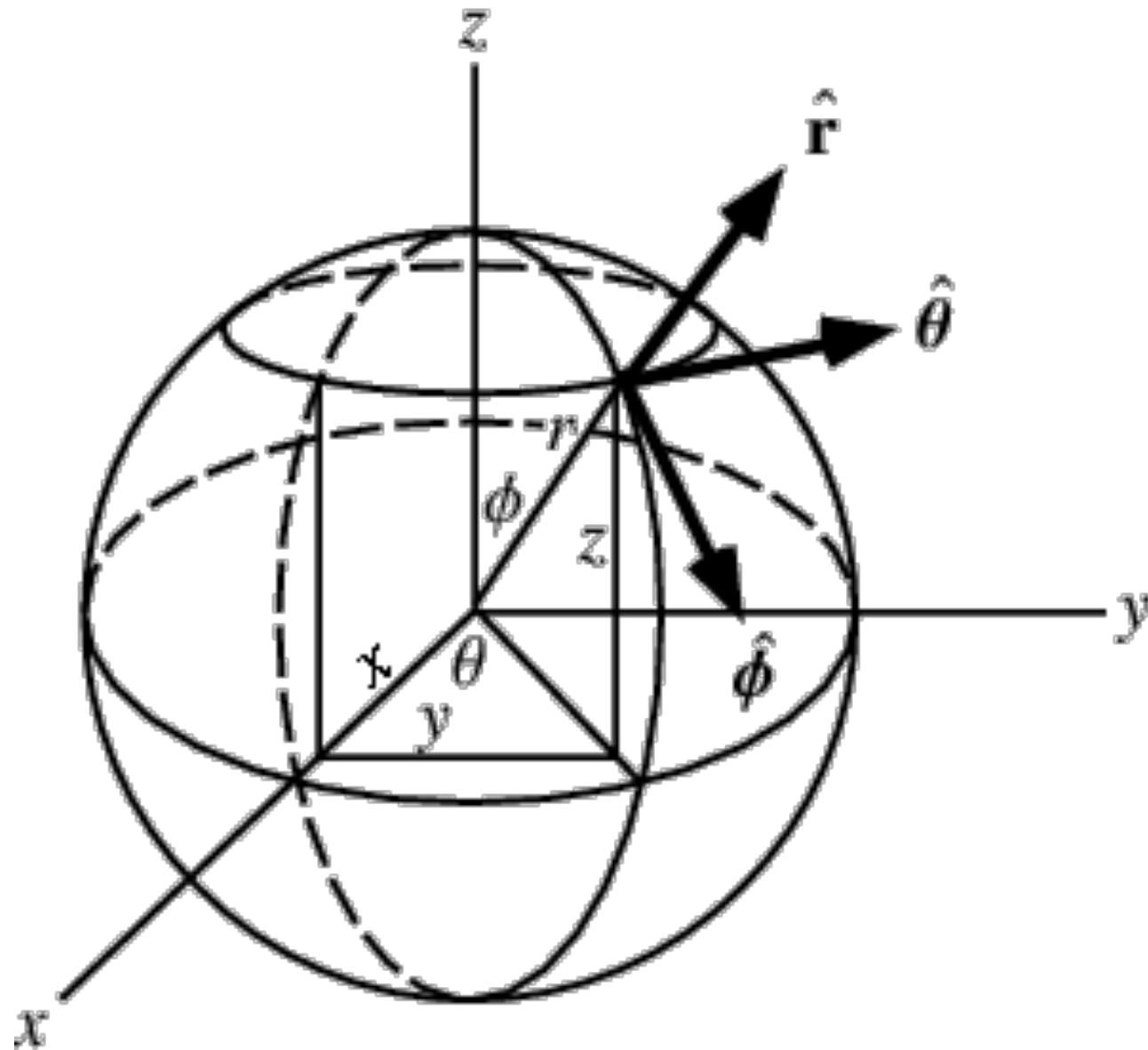


Erste Stufe

3 Dimensional



Zweite Stufe

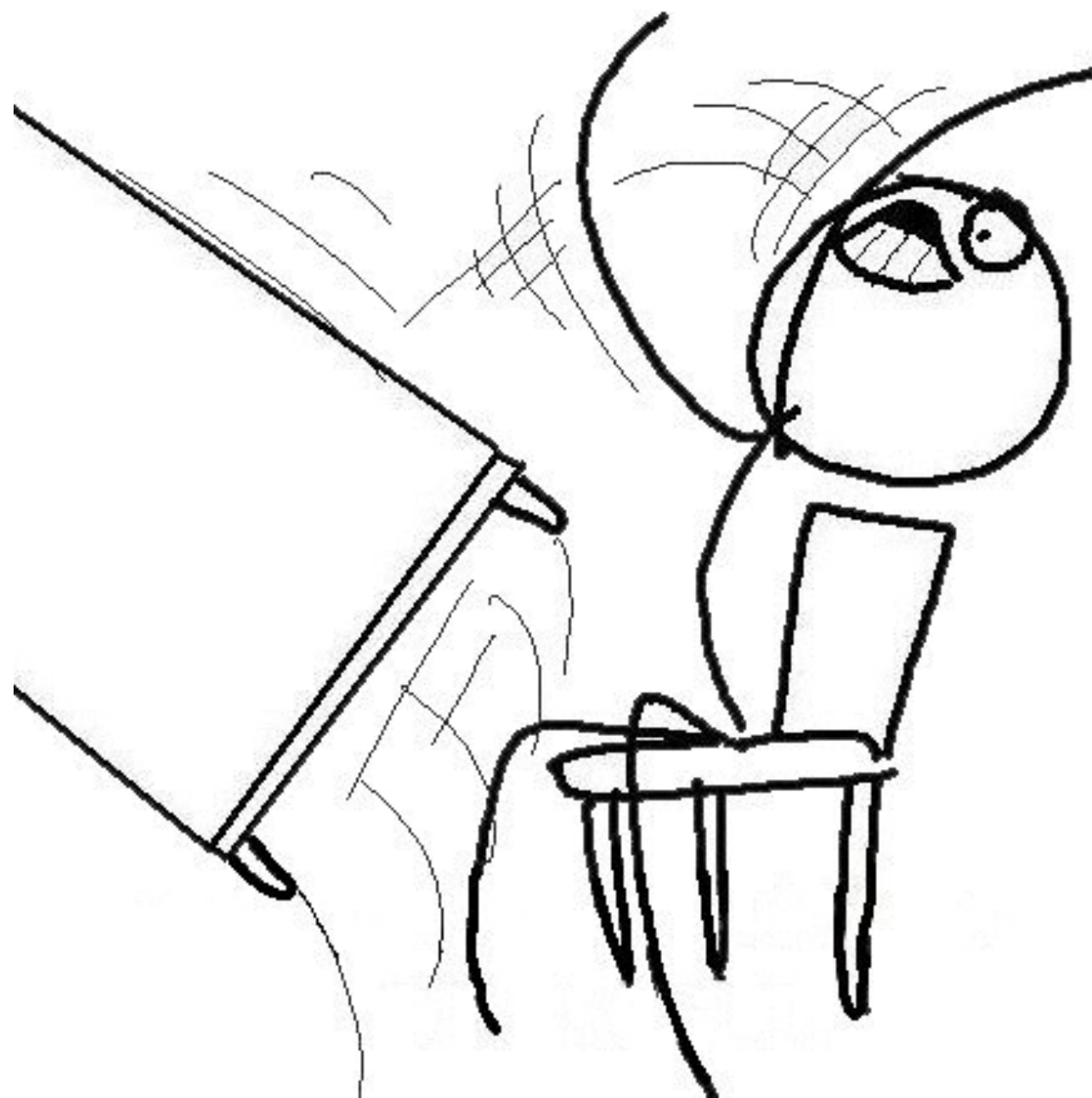


SphericalCoordinates

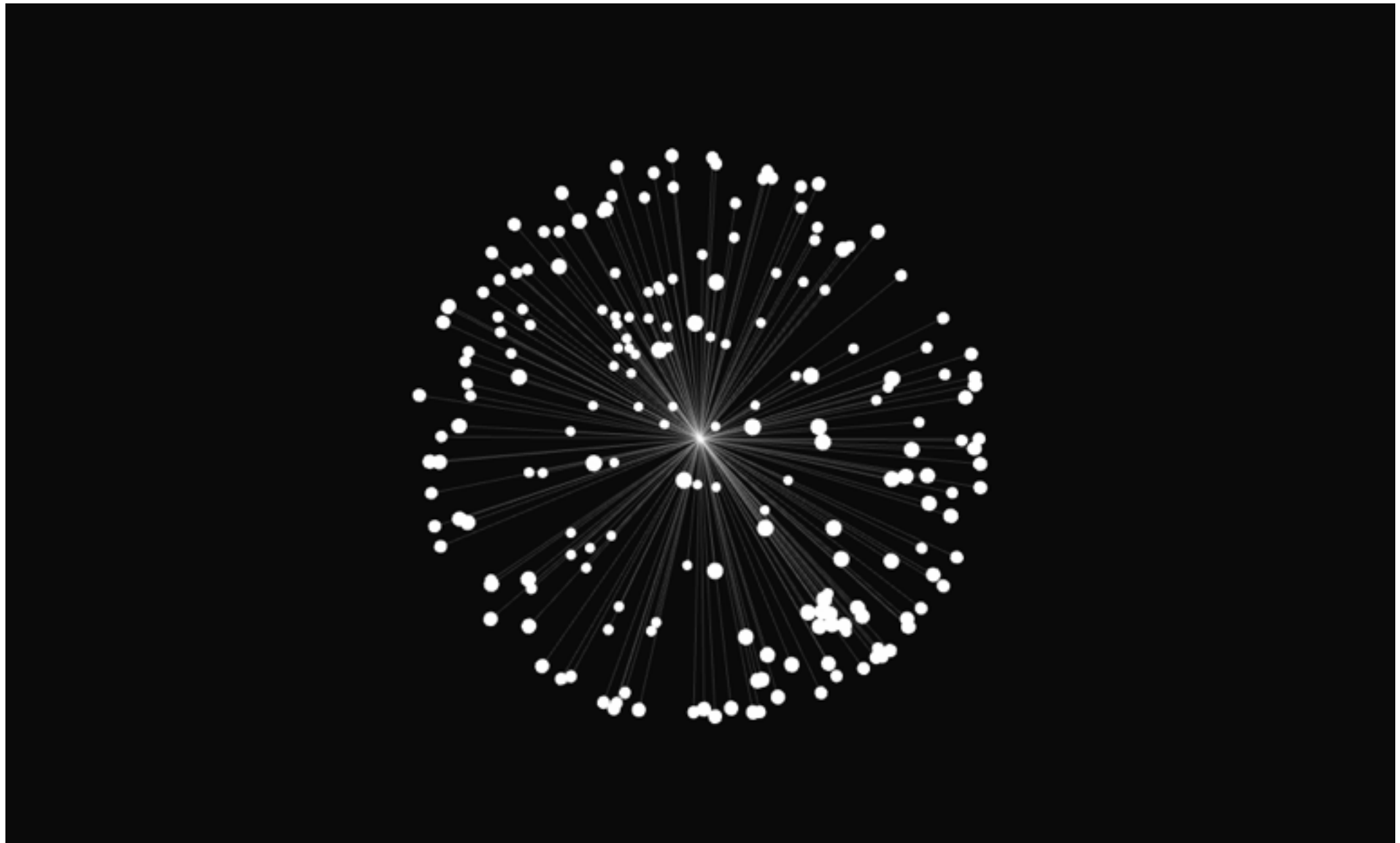
<http://mathworld.wolfram.com/SphericalCoordinates.html>

Cartesian Coordinates

- $x = r \cdot \cos \theta \cdot \sin \Phi$
- $y = r \cdot \sin \theta \cdot \sin \Phi$
- $z = r \cdot \cos \Phi$



- `vector<ofVec2f> point3D;`
- `point3DSetting();`
 - `float _theta = ofRandom(0, TWO_PI);`
 - `float _pi = ofRandom(0, PI);`
 - `float _radius = 200;`
 - `float _x = cos(_theta) * sin(_pi) * _radius;`
 - `float _y = sin(_theta) * sin(_pi) * _radius;`
 - `float _z = sin(_pi) * _radius;`
 - `ofVec3f _point3D_temp = ofVec3f(_x, _y, _z);`
 - `point3D.push_back(_point3D_temp);`



Zweite Stufe