Testy implementacji klasy Mat4, macierzy 4x4.

ADD TEST

#[test]  
fn add\_test() {  
 let mut m = Mat4::identity();  
 m.scale(Vector::new(2.0, 2.0, 2.0));  
 m.translate(Vector::new(1.0, 2.0, 3.0));  
 let mut m2 = Mat4::identity();  
 m2.scale(Vector::new(2.0, 2.0, 2.0));  
 m2.translate(Vector::new(1.0, 2.0, 3.0));  
 m += m2;  
 let result = Mat4 {  
 m: [  
 [4.0, 0.0, 0.0, 2.0],  
 [0.0, 4.0, 0.0, 4.0],  
 [0.0, 0.0, 4.0, 6.0],  
 [0.0, 0.0, 0.0, 2.0],  
 ],  
 };  
 assert\_eq!(m, result);  
}

we have a matrix:

[ 2.000 0.000 0.000 1.000 ]

[ 0.000 2.000 0.000 2.000 ]

[ 0.000 0.000 2.000 3.000 ]

[ 0.000 0.000 0.000 1.000 ]

we add it with a matrix:

[ 2.000 0.000 0.000 1.000 ]

[ 0.000 2.000 0.000 2.000 ]

[ 0.000 0.000 2.000 3.000 ]

[ 0.000 0.000 0.000 1.000 ]

the result is:

[ 4.000 0.000 0.000 2.000 ]

[ 0.000 4.000 0.000 4.000 ]

[ 0.000 0.000 4.000 6.000 ]

[ 0.000 0.000 0.000 2.000 ]

Which is what it should be

SUB TEST

#[test]  
fn sub\_test() {  
 let mut m = Mat4::identity();  
 m.scale(Vector::new(2.0, 2.0, 2.0));  
 m.translate(Vector::new(1.0, 2.0, 3.0));  
 let mut m2 = Mat4::identity();  
 m2.scale(Vector::new(2.0, 2.0, 2.0));  
 m2.translate(Vector::new(1.0, 2.0, 3.0));  
 m -= m2;  
 let result = Mat4 {  
 m: [  
 [0.0, 0.0, 0.0, 0.0],  
 [0.0, 0.0, 0.0, 0.0],  
 [0.0, 0.0, 0.0, 0.0],  
 [0.0, 0.0, 0.0, 0.0],  
 ],  
 };  
 assert\_eq!(m, result);  
}

we have a matrix:

[ 2.000 0.000 0.000 1.000 ]

[ 0.000 2.000 0.000 2.000 ]

[ 0.000 0.000 2.000 3.000 ]

[ 0.000 0.000 0.000 1.000 ]

we add it with a matrix:

[ 2.000 0.000 0.000 1.000 ]

[ 0.000 2.000 0.000 2.000 ]

[ 0.000 0.000 2.000 3.000 ]

[ 0.000 0.000 0.000 1.000 ]

the result is:

[ 0.000 0.000 0.000 0.000 ]

[ 0.000 0.000 0.000 0.000 ]

[ 0.000 0.000 0.000 0.000 ]

[ 0.000 0.000 0.000 0.000 ]

Which is what it should be

INVERSE TEST

#[test]  
fn inverse\_test() {  
 let mut m = Mat4::identity();  
 m.scale(Vector::new(2.0, 2.0, 2.0));  
 m.translate(Vector::new(1.0, 2.0, 3.0));  
 m.inverse();  
 let result = Mat4 {  
 m: [  
 [0.5, 0.0, 0.0, -0.5],  
 [0.0, 0.5, 0.0, -1.0],  
 [0.0, 0.0, 0.5, -1.5],  
 [0.0, 0.0, 0.0, 1.0],  
 ],  
 };  
 assert\_eq!(m, result);  
}

we have a matrix:

[ 2.000 0.000 0.000 1.000 ]

[ 0.000 2.000 0.000 2.000 ]

[ 0.000 0.000 2.000 3.000 ]

[ 0.000 0.000 0.000 1.000 ]

after inverting it we should have this:

[ 0.500 0.000 0.000 -0.500 ]

[ 0.000 0.500 0.000 -1.000 ]

[ 0.000 0.000 0.500 -1.500 ]

[ 0.000 0.000 0.000 1.000 ]

and our matrix is this:

[ 0.500 0.000 0.000 -0.500 ]

[ 0.000 0.500 0.000 -1.000 ]

[ 0.000 0.000 0.500 -1.500 ]

[ 0.000 0.000 0.000 1.000 ]

they are identical

IDENTITY TEST

#[test]  
fn test\_identity() {  
 let m = Mat4::identity();  
  
 let identity = Mat4 {  
 m: [  
 [1.0, 0.0, 0.0, 0.0],  
 [0.0, 1.0, 0.0, 0.0],  
 [0.0, 0.0, 1.0, 0.0],  
 [0.0, 0.0, 0.0, 1.0],  
 ],  
 };  
 assert\_eq!(m, identity);  
}

by calling Mat4::identity() we should have a matrix like this:

[ 1.000 0.000 0.000 0.000 ]

[ 0.000 1.000 0.000 0.000 ]

[ 0.000 0.000 1.000 0.000 ]

[ 0.000 0.000 0.000 1.000 ]

and our matrix is this this:

[ 1.000 0.000 0.000 0.000 ]

[ 0.000 1.000 0.000 0.000 ]

[ 0.000 0.000 1.000 0.000 ]

[ 0.000 0.000 0.000 1.000 ]

they are identical

MULTIPLICATION TEST

#[test]  
fn test\_multiply() {  
 let mut m = Mat4::identity();  
  
 m.scale(Vector::new(3.0, 2.0, 5.0));  
 m.translate(Vector::new(1.0, 2.0, 3.0));  
  
 let mut m2 = Mat4 {  
 m: [  
 [1.0, 2.0, 3.0, 4.0],  
 [5.0, 6.0, 7.0, 8.0],  
 [9.0, 10.0, 11.0, 12.0],  
 [13.0, 14.0, 15.0, 16.0],  
 ],  
 };  
 m.multiply(&m2);  
  
 let result = Mat4 {  
 m: [  
 [16.0, 20.0, 24.0, 28.0],  
 [36.0, 40.0, 44.0, 48.0],  
 [84.0, 92.0, 100.0, 108.0],  
 [13.0, 14.0, 15.0, 16.0],  
 ],  
 };  
  
 assert\_eq!(m, result);  
}

we have two matrices:

[ 3.000 0.000 0.000 1.000 ]

[ 0.000 2.000 0.000 2.000 ]

[ 0.000 0.000 5.000 3.000 ]

[ 0.000 0.000 0.000 1.000 ]

and

[ 1.000 2.000 3.000 4.000 ]

[ 5.000 6.000 7.000 8.000 ]

[ 9.000 10.000 11.000 12.000 ]

[ 13.000 14.000 15.000 16.000 ]

after multiplying them we should have this:

[ 16.000 20.000 24.000 28.000 ]

[ 36.000 40.000 44.000 48.000 ]

[ 84.000 92.000 100.000 108.000 ]

[ 13.000 14.000 15.000 16.000 ]

and our matrix is this this:

[ 16.000 20.000 24.000 28.000 ]

[ 36.000 40.000 44.000 48.000 ]

[ 84.000 92.000 100.000 108.000 ]

[ 13.000 14.000 15.000 16.000 ]

they are identical

ROTATION TEST

#[test]  
fn test\_rotate() {  
 let mut m = Mat4::identity();  
 m.rotate(as\_radians(90.0), Vector::new(0.0, 0.0, 1.0));  
 let result = Mat4 {  
 m: [  
 [0.0, -1.0, 0.0, 0.0],  
 [1.0, 0.0, 0.0, 0.0],  
 [0.0, 0.0, 1.0, 0.0],  
 [0.0, 0.0, 0.0, 1.0],  
 ],  
 };  
 assert\_eq!(m, result);  
  
 m = Mat4::identity();  
  
 m.rotate(as\_radians(90.0), Vector::new(0.0, 1.0, 0.0));  
 let result = Mat4 {  
 m: [  
 [0.0, 0.0, 1.0, 0.0],  
 [0.0, 1.0, 0.0, 0.0],  
 [-1.0, 0.0, 0.0, 0.0],  
 [0.0, 0.0, 0.0, 1.0],  
 ],  
 };  
 assert\_eq!(m, result);  
}

we have a matrix:

[ 1.000 0.000 0.000 0.000 ]

[ 0.000 1.000 0.000 0.000 ]

[ 0.000 0.000 1.000 0.000 ]

[ 0.000 0.000 0.000 1.000 ]

after rotating by 90deg in z axis, we should have this:

[ 0.000 -1.000 0.000 0.000 ]

[ 1.000 0.000 0.000 0.000 ]

[ 0.000 0.000 1.000 0.000 ]

[ 0.000 0.000 0.000 1.000 ]

Now we have a matrix:

[ 1.000 0.000 0.000 0.000 ]

[ 0.000 1.000 0.000 0.000 ]

[ 0.000 0.000 1.000 0.000 ]

[ 0.000 0.000 0.000 1.000 ]

after rotating by 90deg in y axis, we should have this:

[ 0.000 0.000 1.000 0.000 ]

[ 0.000 1.000 0.000 0.000 ]

[ -1.000 0.000 0.000 0.000 ]

[ 0.000 0.000 0.000 1.000 ]

and our matrix is this:

[ 0.000 0.000 1.000 0.000 ]

[ 0.000 1.000 0.000 0.000 ]

[ -1.000 0.000 0.000 0.000 ]

[ 0.000 0.000 0.000 1.000 ]

they are identical

SCALE TEST

#[test]  
fn test\_scale() {  
 let mut m = Mat4::identity();  
 m.scale(Vector::new(1.0, 2.0, 3.0));  
 let result = Mat4 {  
 m: [  
 [1.0, 0.0, 0.0, 0.0],  
 [0.0, 2.0, 0.0, 0.0],  
 [0.0, 0.0, 3.0, 0.0],  
 [0.0, 0.0, 0.0, 1.0],  
 ],  
 };  
 assert\_eq!(m, result);  
}

we have a matrix:

[ 1.000 0.000 0.000 0.000 ]

[ 0.000 1.000 0.000 0.000 ]

[ 0.000 0.000 1.000 0.000 ]

[ 0.000 0.000 0.000 1.000 ]

after scaling it we should have this:

[ 1.000 0.000 0.000 0.000 ]

[ 0.000 2.000 0.000 0.000 ]

[ 0.000 0.000 3.000 0.000 ]

[ 0.000 0.000 0.000 1.000 ]

and our matrix is this this:

[ 1.000 0.000 0.000 0.000 ]

[ 0.000 2.000 0.000 0.000 ]

[ 0.000 0.000 3.000 0.000 ]

[ 0.000 0.000 0.000 1.000 ]  
they are identical

TRANSLATION TEST

#[test]  
fn test\_translate() {  
 let mut m = Mat4::identity();  
 m.translate(Vector::new(1.0, 2.0, 3.0));  
 let result = Mat4 {  
 m: [  
 [1.0, 0.0, 0.0, 1.0],  
 [0.0, 1.0, 0.0, 2.0],  
 [0.0, 0.0, 1.0, 3.0],  
 [0.0, 0.0, 0.0, 1.0],  
 ],  
 };  
 assert\_eq!(m, result);  
}

we have a matrix:

[ 1.000 0.000 0.000 1.000 ]

[ 0.000 1.000 0.000 2.000 ]

[ 0.000 0.000 1.000 3.000 ]

[ 0.000 0.000 0.000 1.000 ]

after translating it we should have this:

[ 1.000 0.000 0.000 1.000 ]

[ 0.000 1.000 0.000 2.000 ]

[ 0.000 0.000 1.000 3.000 ]

[ 0.000 0.000 0.000 1.000 ]

and our matrix is this this:

[ 1.000 0.000 0.000 1.000 ]

[ 0.000 1.000 0.000 2.000 ]

[ 0.000 0.000 1.000 3.000 ]

[ 0.000 0.000 0.000 1.000 ]

TRANSPOSE TEST

#[test]  
fn transpose\_test() {  
 let mut m = Mat4::identity();  
 m.scale(Vector::new(2.0, 2.0, 2.0));  
 m.translate(Vector::new(1.0, 2.0, 3.0));  
 m.transpose();  
 let result = Mat4 {  
 m: [  
 [2.0, 0.0, 0.0, 0.0],  
 [0.0, 2.0, 0.0, 0.0],  
 [0.0, 0.0, 2.0, 0.0],  
 [1.0, 2.0, 3.0, 1.0],  
 ],  
 };  
 assert\_eq!(m, result);  
}

we have a matrix:

[ 2.000 0.000 0.000 1.000 ]

[ 0.000 2.000 0.000 2.000 ]

[ 0.000 0.000 2.000 3.000 ]

[ 0.000 0.000 0.000 1.000 ]

after transposing it we should have this:

[ 2.000 0.000 0.000 0.000 ]

[ 0.000 2.000 0.000 0.000 ]

[ 0.000 0.000 2.000 0.000 ]

[ 1.000 2.000 3.000 1.000 ]

and our matrix is this:

[ 2.000 0.000 0.000 0.000 ]

[ 0.000 2.000 0.000 0.000 ]

[ 0.000 0.000 2.000 0.000 ]

[ 1.000 2.000 3.000 1.000 ]

they are identical