Report of Computer Animation Homework 1

Wenli Zhang

http://zhangwenli.com

5090379039 OviliaZhang@gmail.com

Abstract

In this project, I implemented animations of an artillery simulator and a spring-mass simulator, both of which were implemented using Euler's method, Mid-point and forth-order Runge-Kutta. UI is made to be self-explanatory and easy to use. Analysis of different algorithms and conclusions are made in the final part.

How to run

I used WebGL to do the rendering part as I'm most familiar with it and it's more efficient to use WebGL then OpenGL since by using third-party libraries to do other jobs like event handling and UI control, I can have more time to focus on the animation algorithm part.

You can access the two animations at the following URL:

http://zhangwenli.com/ComupterAnimation/artillery.html

http://zhangwenli.com/ComputerAnimation/spring.html

Or, if you have a server, you may put the *src* folder under your server and run in modern web browsers like Chrome and Firefox.

Code is also available at https://github.com/Ovilia/ComputerAnimation.

How to use

1. Spring-mass Simulator

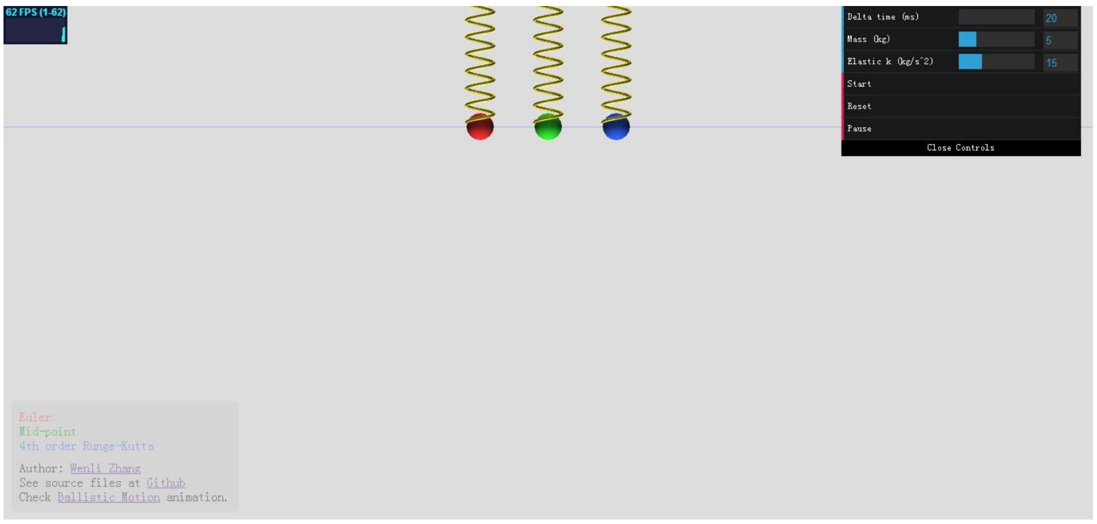


Figure 1 Initial scene of spring-mass simulator

Left-top: FPS information
Right-top: Parameters
Left-bottom: Additional information

Note that initial release position is where the spring has no deformation.
Three spheres are of different colors. The red one uses Euler’s method, the green one Mid-point, and the blue one forth-order Runge-Kutta.
Click *Start* on the right-top to release.
To make the positions of three spheres clearer, there are three lines of respective colors to represent the center positions of the spheres, as shown in Figure 2.
There are also two lines indicating the top and bottom positions, which serve as a clue of position change during different periods of vibration.

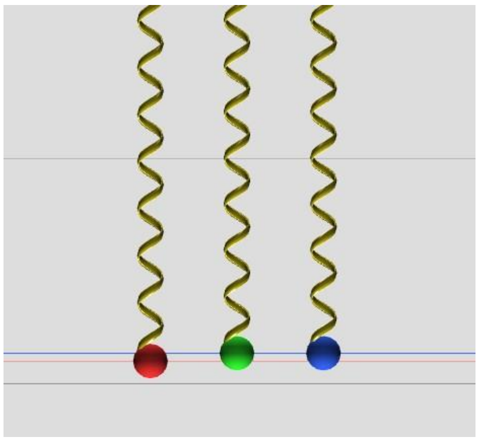
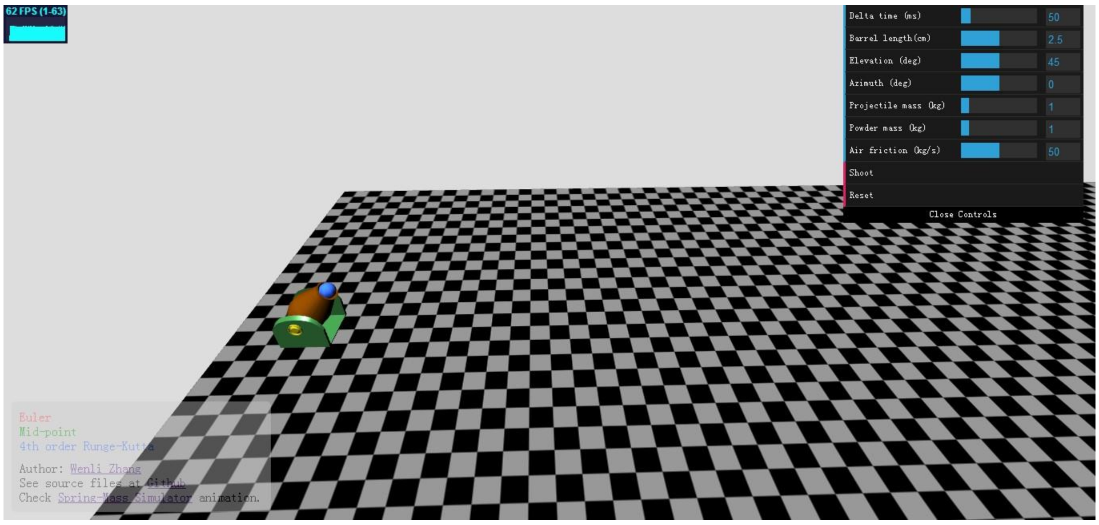


Figure 2 Lines as hint of position

2. Artillery Simulator


62 FPS (1-63)

Delta time (ms)	50
Barrel length(cm)	2.5
Elevation (deg)	45
Azimuth (deg)	0
Projectile mass (kg)	1
Powder mass (kg)	1
Air friction (kg/s)	50

Shoot
Reset
Close Controls

Euler
Mid-point
4th order Runge-Kutta

Author: Wenli Zhang
See source files at Github
Check Spring-Mass Simulator animation.

Figure 3 Initial scene of artillery simulator
Left-mouse drag to rotate the scene
Middle-mouse drag to move the scene
Scroll to zoom in or out the scene

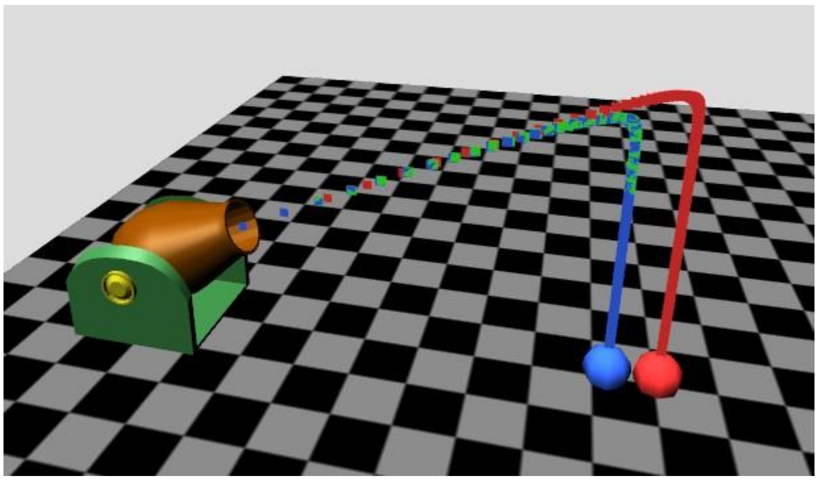


Figure 4 Projectiles with different colors

Three projectiles of different colors are implemented using different algorithms, as shown in Figure 4. Mid-point and forth-order Runge-Kutta usually produce similar results, so the green and blue projectiles are almost at the same position.

Algorithms

Euler's method is the easiest to implement but the least accurate in most cases. It assumes speed doesn't change in a short period of time and use the original speed as average speed in this period of time. So, when step time is large enough, Euler's method can sometimes produce odd results. Mid-point and forth-order Runge-Kutta use Euler's result as a clue and optimize by calculating intermediate state.
Implementation can be found in *_logic.js files.
See Analyses part for comparation of three algorithms.

Analyses

- **Spring-mass Simulator**

1. **Compare results of different algorithms with mass = 5kg and k = 15kg/s^2**
(Delta time: 50ms, mass: 5kg, k: 15kg/s^2)

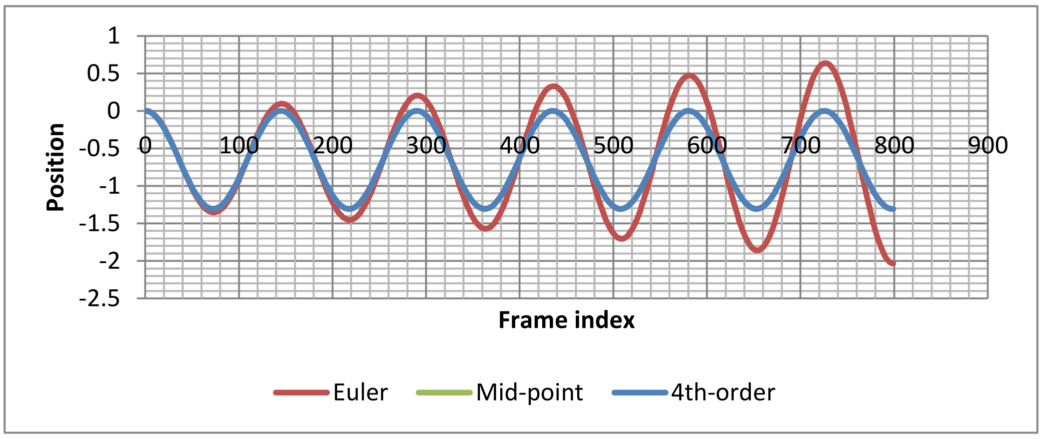


Figure 5 Different algorithms

As we can see from Figure 5, results of Mid-point and forth-order are quite similar to each other and share constant amplitude during different periods, while amplitude of Euler's method increases as time goes by.
There are even positions greater than zero, which means positions are higher than the origin. This is impossible both in theory and in fact. So Euler's method is not accurate in this case.

2. Compare results of different parameters

(Delta time: 50ms)

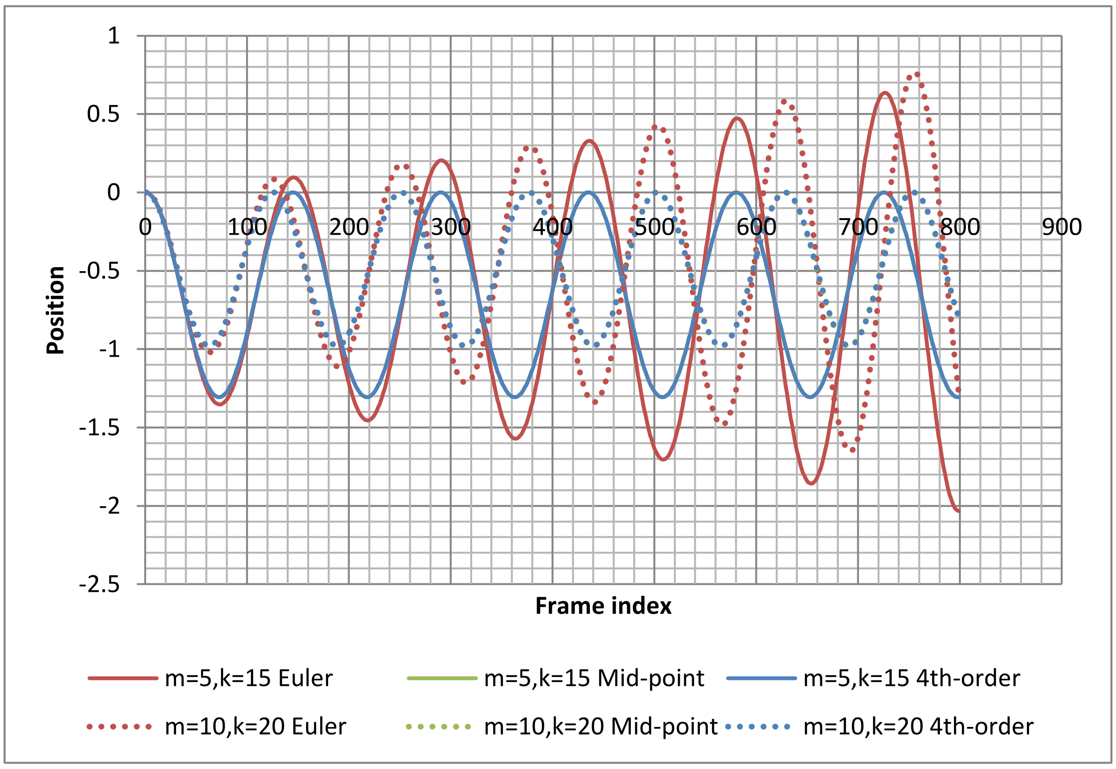


Figure 6 Different mass and k

Different mass and k can change the period and amplitude of the vibration. Mid-point and forth-order Runge-Kutta have good perform over different mass and k values, while amplitude increases during time with Euler's method.

3. Compare results of different step time

(mass: 5kg, k: 15kg/s^2)

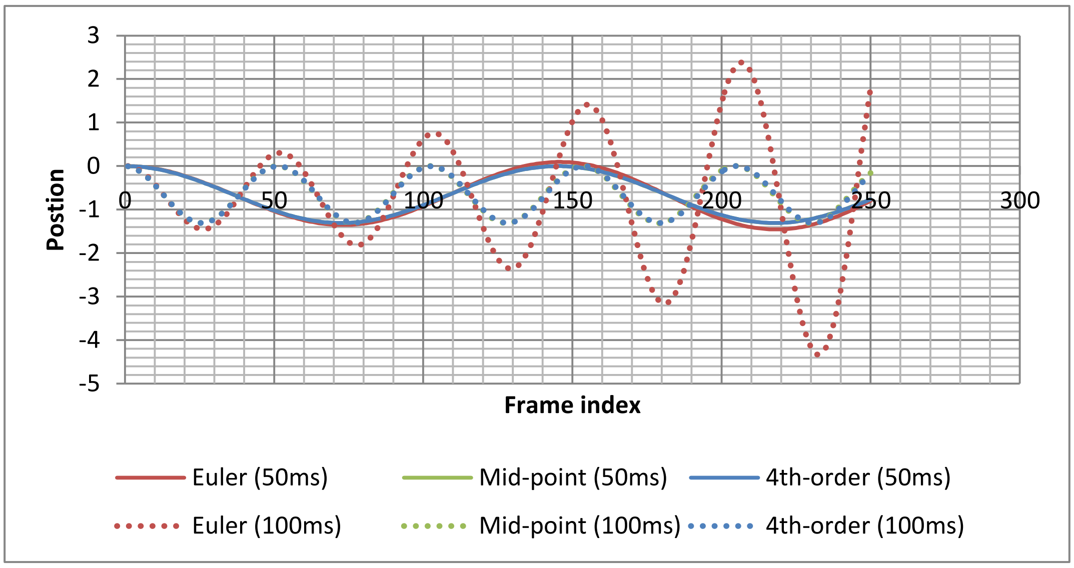


Figure 7 Different step time

As we can see in Figure 7, results of Euler's method are closer to Mid-point and forth-order when step time is shorter.

● **Artillery Simulator**

1. Compare results of different algorithms with default parameters

(Delta time: 50ms, Barrel length: 2.5m, Elevation: 45deg, Azimuth: 0deg, Projectile mass: 1kg, Powder mass: 1kg, Air friction: 50kg/s)

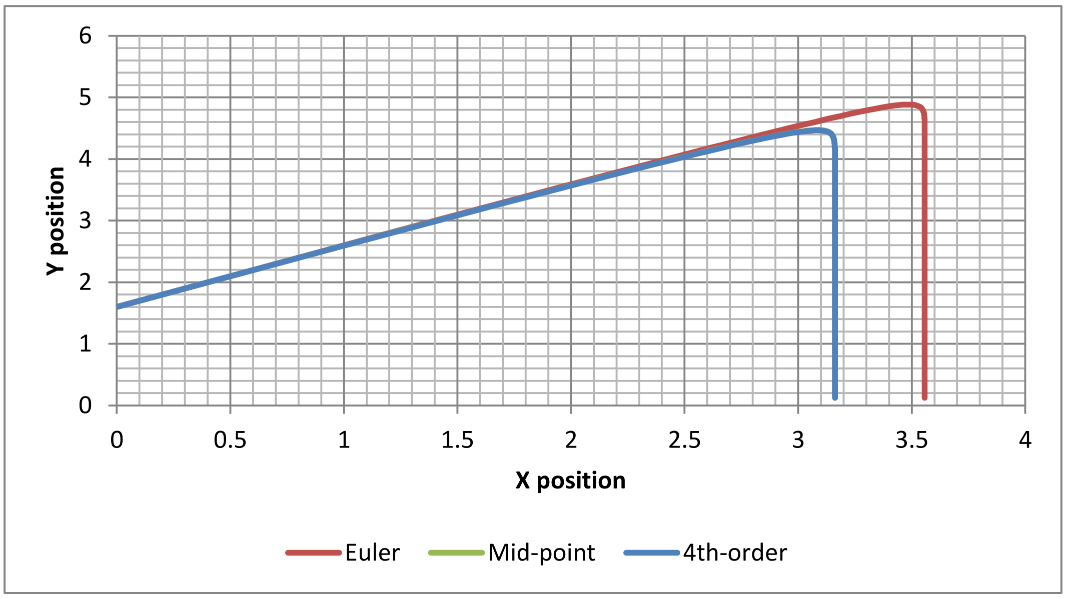


Figure 8 Position calculated using different algorithms with default parameters

Mid-point series is very close to 4th-order series, so it's hidden behind 4th-order in the above chat. From this chat, we can see that results of Euler's method are similar with Mid-point and 4th-order when speed changes smoothly, while when there's a rapid change in speed, Euler's method produces a quite different result.

2. Compare results of different algorithms with different delta time

(Delta time: [50ms, 75ms, 100ms], Barrel length: 2.5m, Elevation: 45deg, Azimuth: 0deg, Projectile mass: 1kg, Powder mass: 1kg, Air friction: 50kg/s)

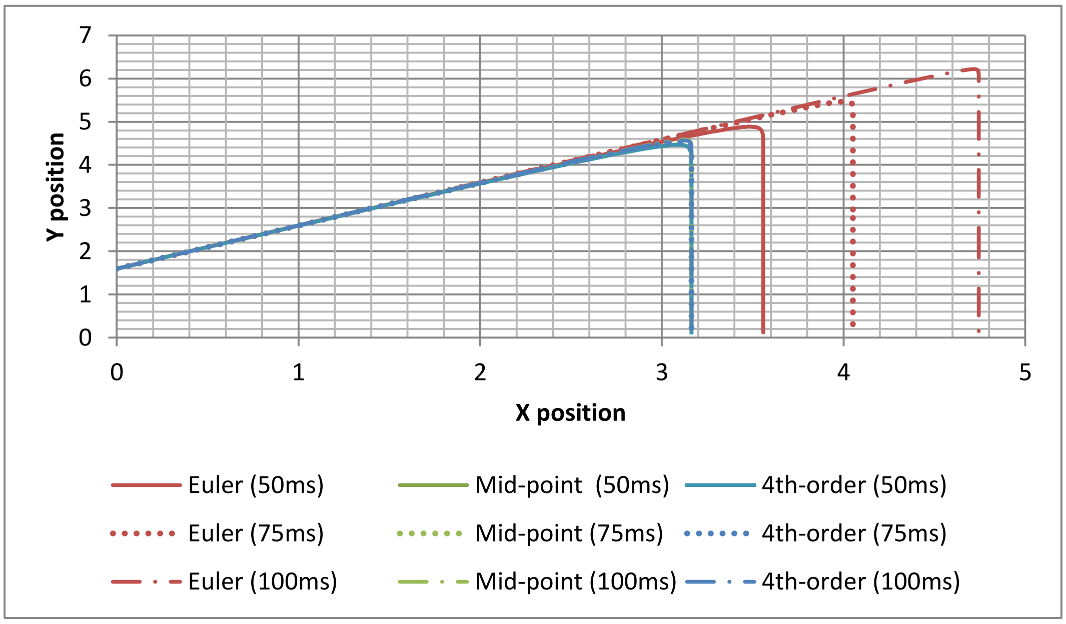


Figure 9 Position calculated using different algorithms with different step time

From Figure 9, we know that results of Euler's method change largely as step time changes, and it is closer to Mid-point's and 4th-order's results when step time is smaller. Results of mid-point and 4th-order almost don't change as step time changes.

3. Compare results of different algorithms with different air friction

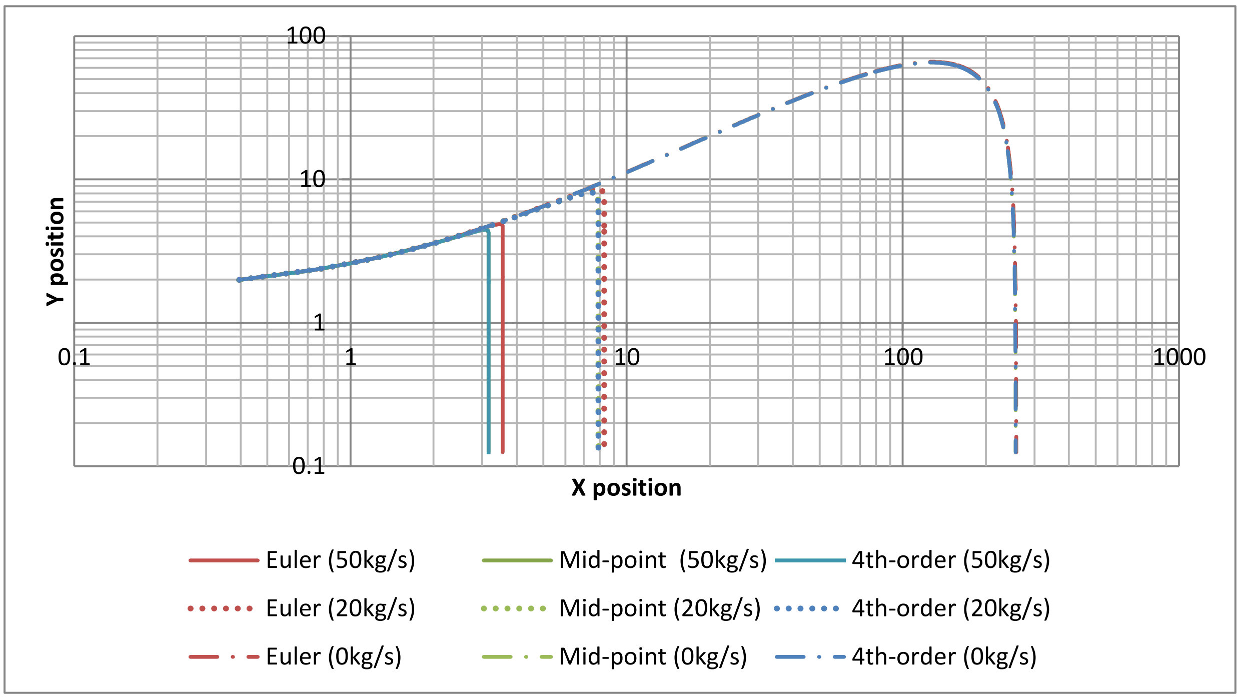


Figure 10 Position calculated using different algorithms with different air friction

Both axes in Figure 10 are logarithmic scaled to see more clearly about the difference between results of three algorithms. We can see that results of all algorithms are more similar to each other when there’s less air friction. This is probably because air friction is related to speed, and even a small distinction in speed will cause a large difference in force and then accumulate to a significant distinction of position.

Conclusion

From the analyses above, we can see that Mid-point and forth-order Runge-Kutta are more accurate and stable then Euler, while the former two show little difference in accuracy or stableness in most cases.
But it doesn’t necessary mean that Mid-point and forth-order Runge-Kutta are better than Euler. Euler is more efficient since less computation is needed in each frame and performs well when the step time is short enough. So it depends on the requirements when choosing a suitable algorithm.

Notice

Files under model folder are models made by myself.
Files under lib folder are third-party libraries.

- jquery-1.8.2.js simplifies developing with JavaScript.
- stats.min.js is for computing FPS on the top-left of the screen.
- DAT.GUI is for the UI control on the top-right of the screen.
- Other files under lib folder are for THREE.js, which simplifies developing with WebGL.

2013.5.11