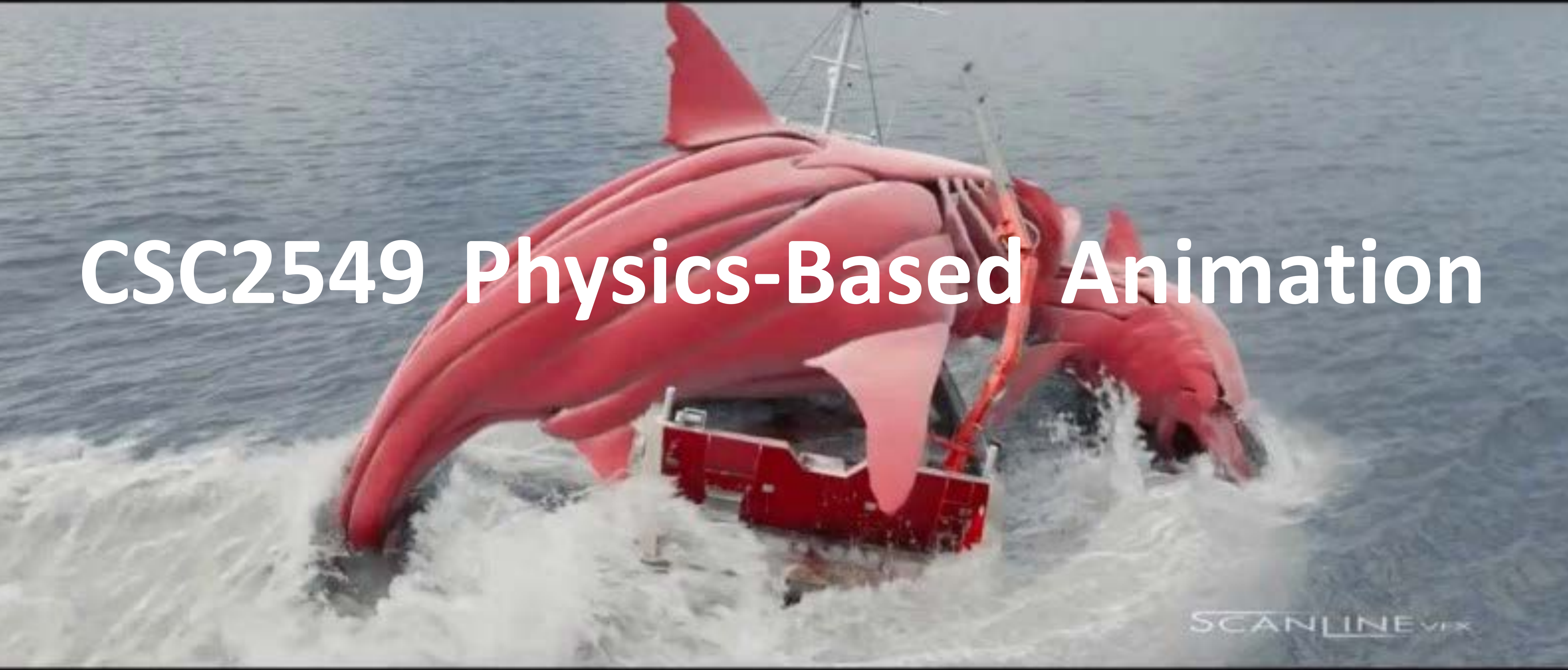
CSC2549 Physics-Based Animation



The Meg | Scanline VFX

Last Week: Introduction and Springs

Today: Time Integration

Reminders

Website:

https://github.com/dilevin/CSC2549-physics-based-animation

Bulletin Board:

https://bb-2019-09.teach.cs.toronto.edu/c/csc2549

MarkUs:

https://markus.teach.cs.toronto.edu/csc2549-2019-09/

Contact:

diwlevin@cs.toronto.edu

More Reminders

Assignment #1 is due next Friday

https://github.com/dilevin/CSC2549-a1-mass-spring-1d

Assignment #2 will be posted tomorrow

Graphics Reading Group

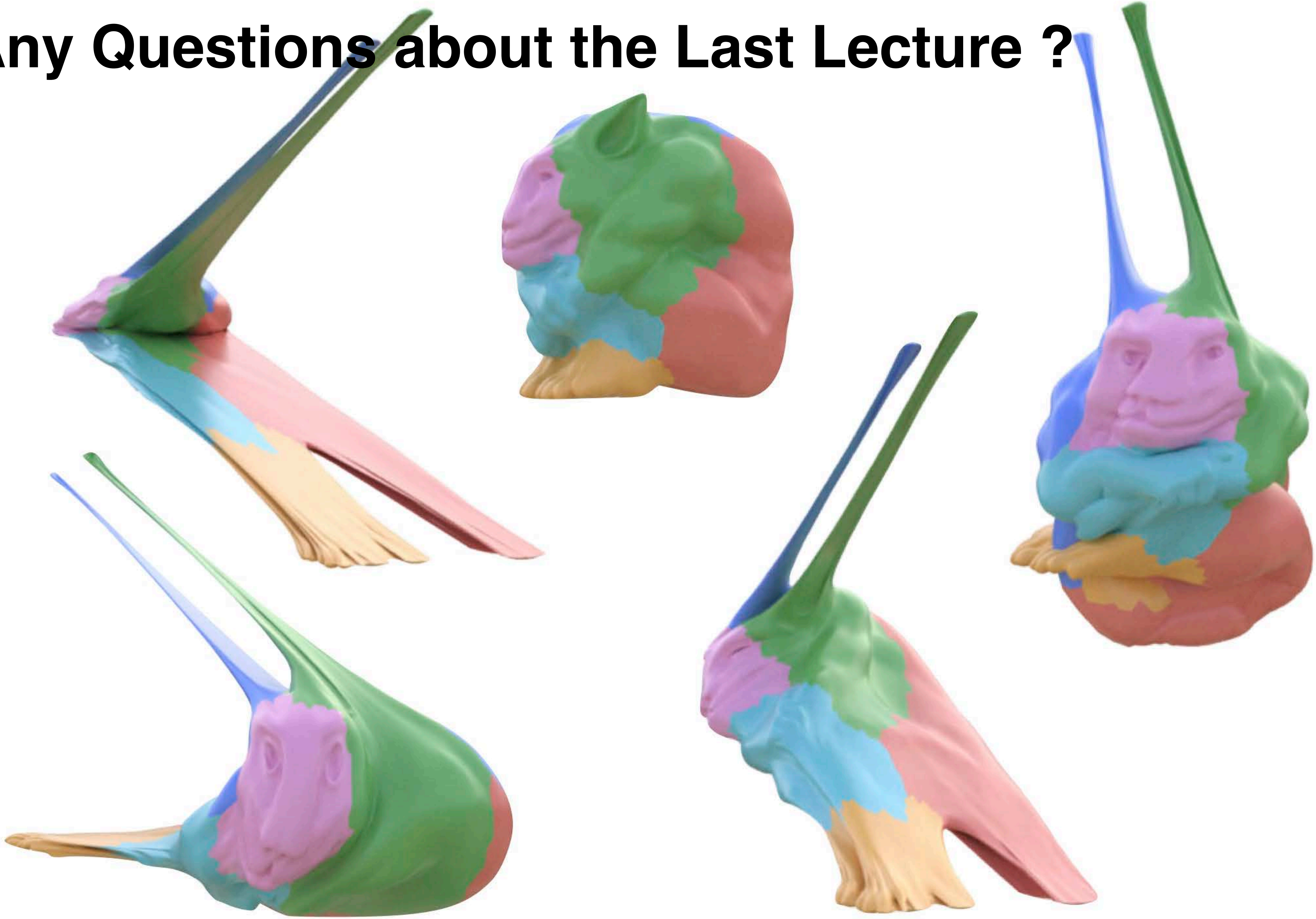
Seminar Room in BA5166 (Dynamic Graphics Project)

Wednesdays 11am

Today

1. Questions about the last lecture
2. Introduction to Time Integration
3. Algorithms
    1. Forward Euler Time Integration
    2. Runge-Kutta Time Integration
    3. Backward (Implicit) Euler Time Integration
    4. Symplectic Euler Integration
4. Some C++ Review

Any Questions about the Last Lecture ?



Fun Math Question

Use the calculus of variations to show that the shortest distance between two points is a straight line

Time Integration

Input: Ordinary Differential Equation: $\ddot{\mathbf{q}} = \mathbf{f}\left(\mathbf{q}, \dot{\mathbf{q}}\right)$

Output: Discrete Update Equation

$$\mathbf{q}^{t+1} = \mathbf{f}\left(\mathbf{q}^{t}, \mathbf{q}^{t+1}, \ldots, \dot{\mathbf{q}}^{t}, \dot{\mathbf{q}}^{t+1}, \ldots\right)$$

The Coupled First Order System

An Illustrative Example

Types of Time Integration

Explicit: Next time step can be computed entirely using values from the current time step

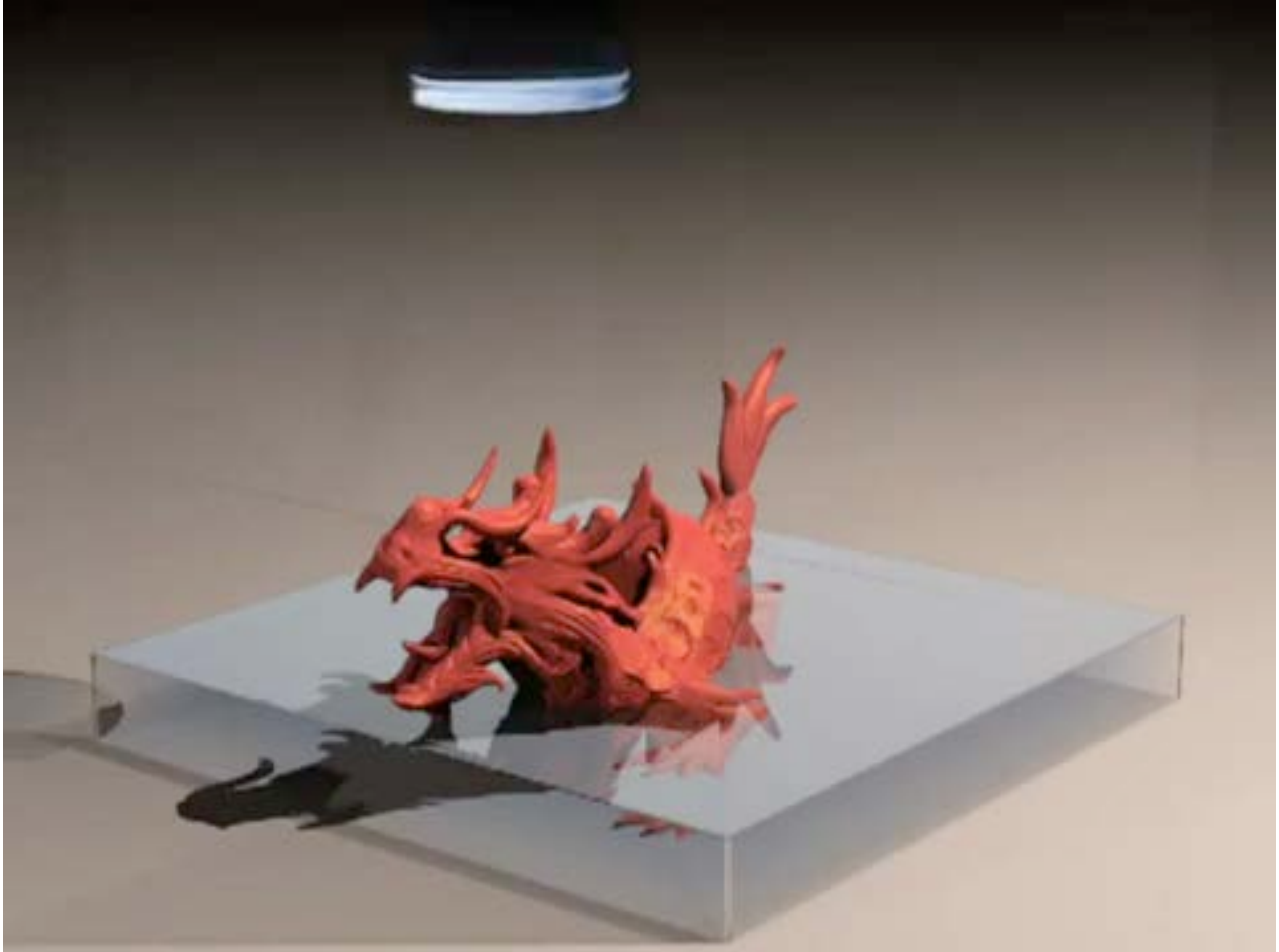
Implicit: Next time step is computed using values from the future!

Concerns

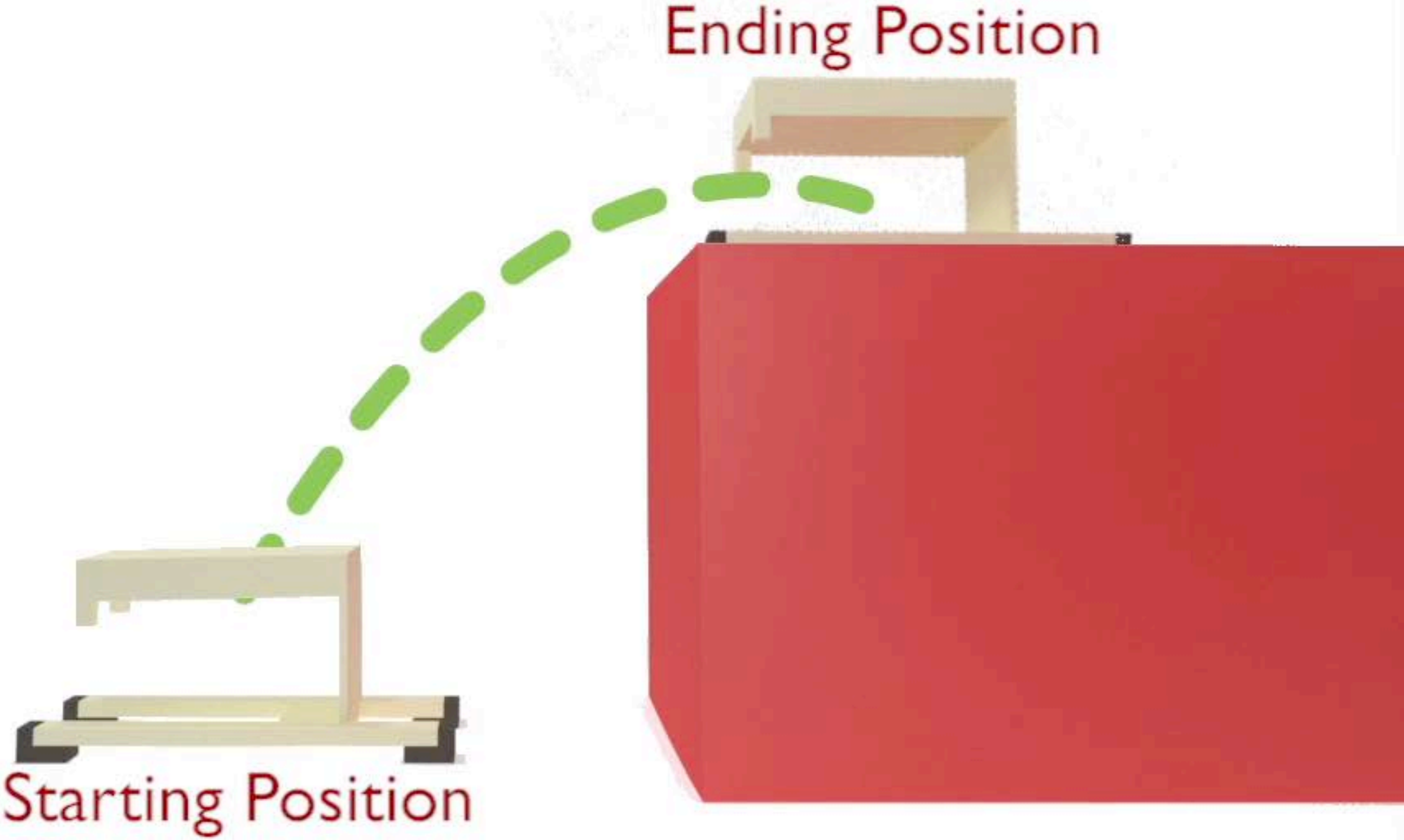
1. Stability

2. Accuracy

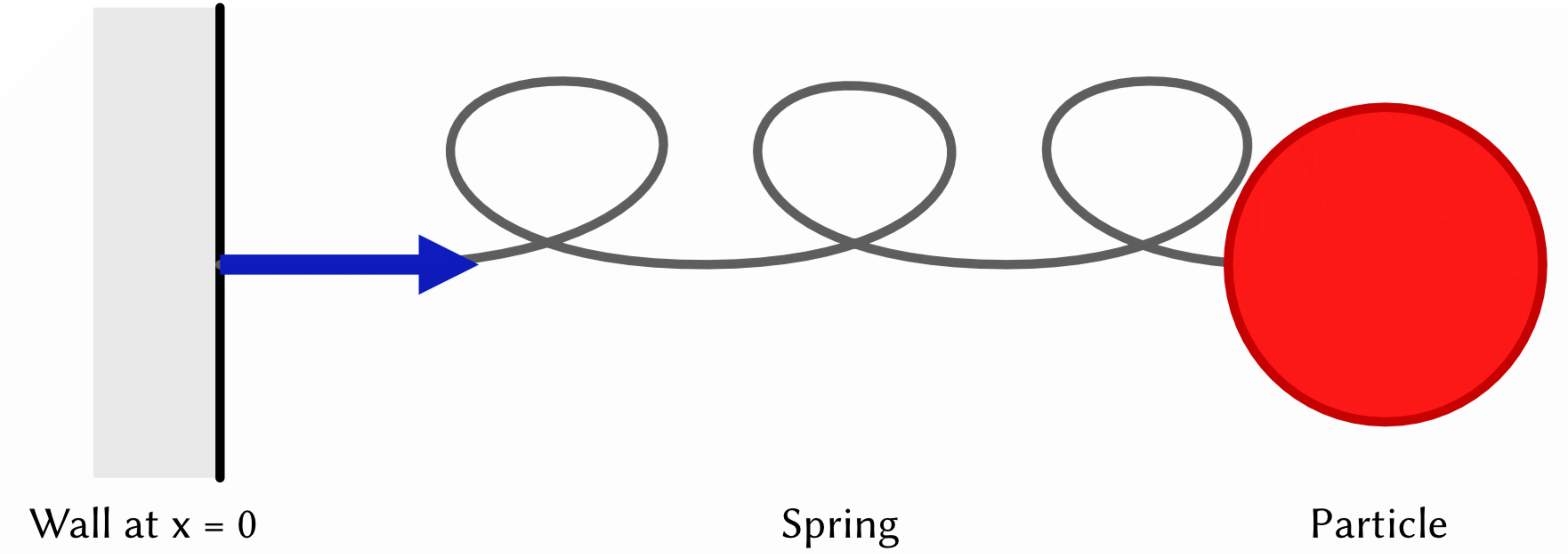
3. Performance



Results



For Assignment 1



Forward-Euler Time Discretization

Stability Analysis

Runge-Kutta

Explicit, multi-step method

Use multiple function evaluations to get a more accurate time step

Backward (Implicit) Euler

Stability Analysis

Symplectic Euler

Symplectic Euler comes from a discrete variational principle

Last class we wrote down the continuous version of the Principle of Least Action, derived the E-L Equations and now we are discretizing these equations

What if we discretize the variational principle ?

Discrete Variational Principle

Discrete Euler-Lagrangre Equations

Some C++ Review

```
#include <Eigen/Dense>

//Input:
//  q - generalized coordiantes for the mass-spring system
//  qdot - generalized velocity for the mass spring system
//  dt - the time step in seconds
//  mass - the mass
//  force(q, qdot) - a function that computes the force acting on the mass as a function. This takes q and qdot as parameters.
//  stiffness(q, qdot) - a function that computes the stiffness (negative second derivative of the potential energy). This takes q and qd
//Output:
//  q - set q to the updated generalized coordinate using Backward Euler time integration
//  qdot - set qdot to the updated generalized velocity using Backward Euler time integration

template<typename FORCE, typename STIFFNESS>
inline void backward_euler(Eigen::VectorXd &q, Eigen::VectorXd &qdot, double dt, double mass,  FORCE &force, STIFFNESS &stiffness) {


}
```

Next Week:

3D Mass-Spring Systems

Final Project Info