COMPSCI 715 Part 2

Lecture 4 - GPGPU Programming

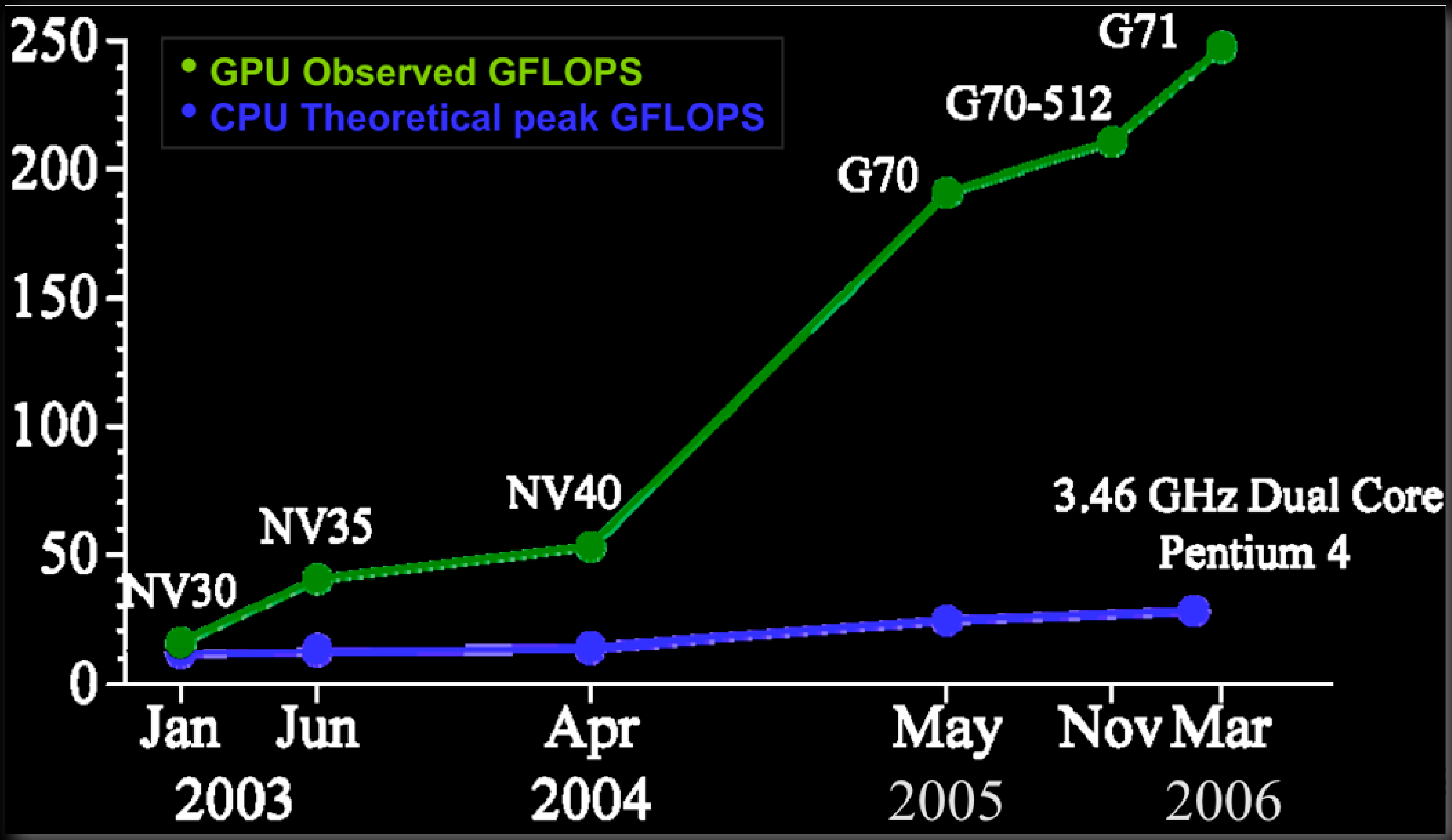
GPGPU

- General Purpose GPU Programming
- New libraries / techniques for using the graphics card for AI, physics and non-game related processing

Why use the GPU?

- Perfect for applications with high degree of parallelization:
  - Image processing
  - Visualization
  - Scientific Computing
  - Medical Imaging (MRI etc)
  - Physics models

CPU vs. GPU



Source: NVIDIA

Problem matching

- G70 chipset:
  - NVIDIA Geforce 7800
  - 24 pixel pipelines
  - 1000s of threads
  - Huge memory bandwidth
- Simple physics:
  - 1000s of collision operations per timestep
  - All relatively independent

Decision Heuristic

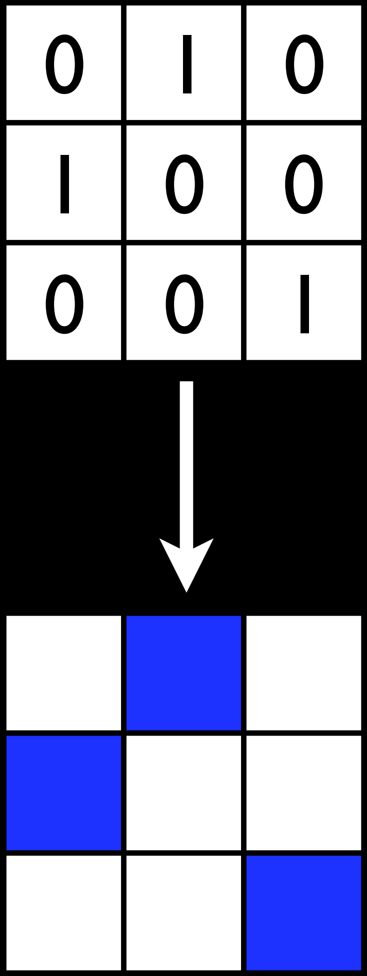
- Arithmetic Intensity = operations / memory lookup
- GPUs work well when intensity is high

Techniques

- Map
  - Operations applied to each element of stream
- Filter
  - Decide which elements to use and remove others
- Gather
- Scatter

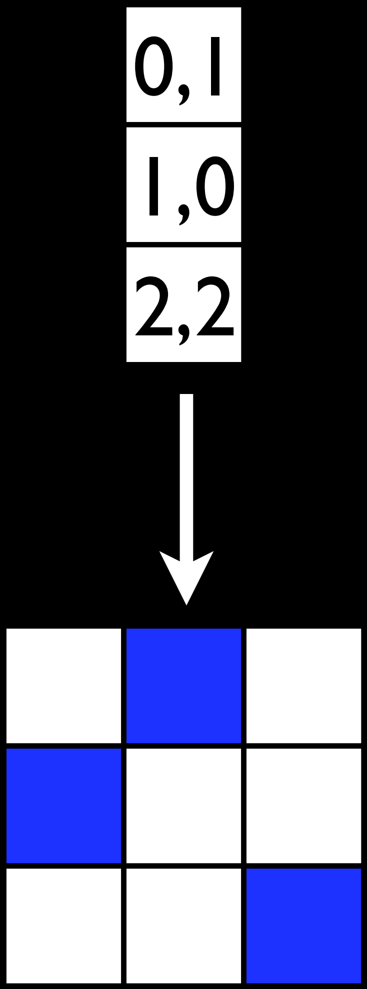
Gather

- Allow the gathering of information from different nodes on a grid
- i.e.
  - Precompute array of elements to calculate into a texture
  - Branch / don't calculate when value is 0



Scatter

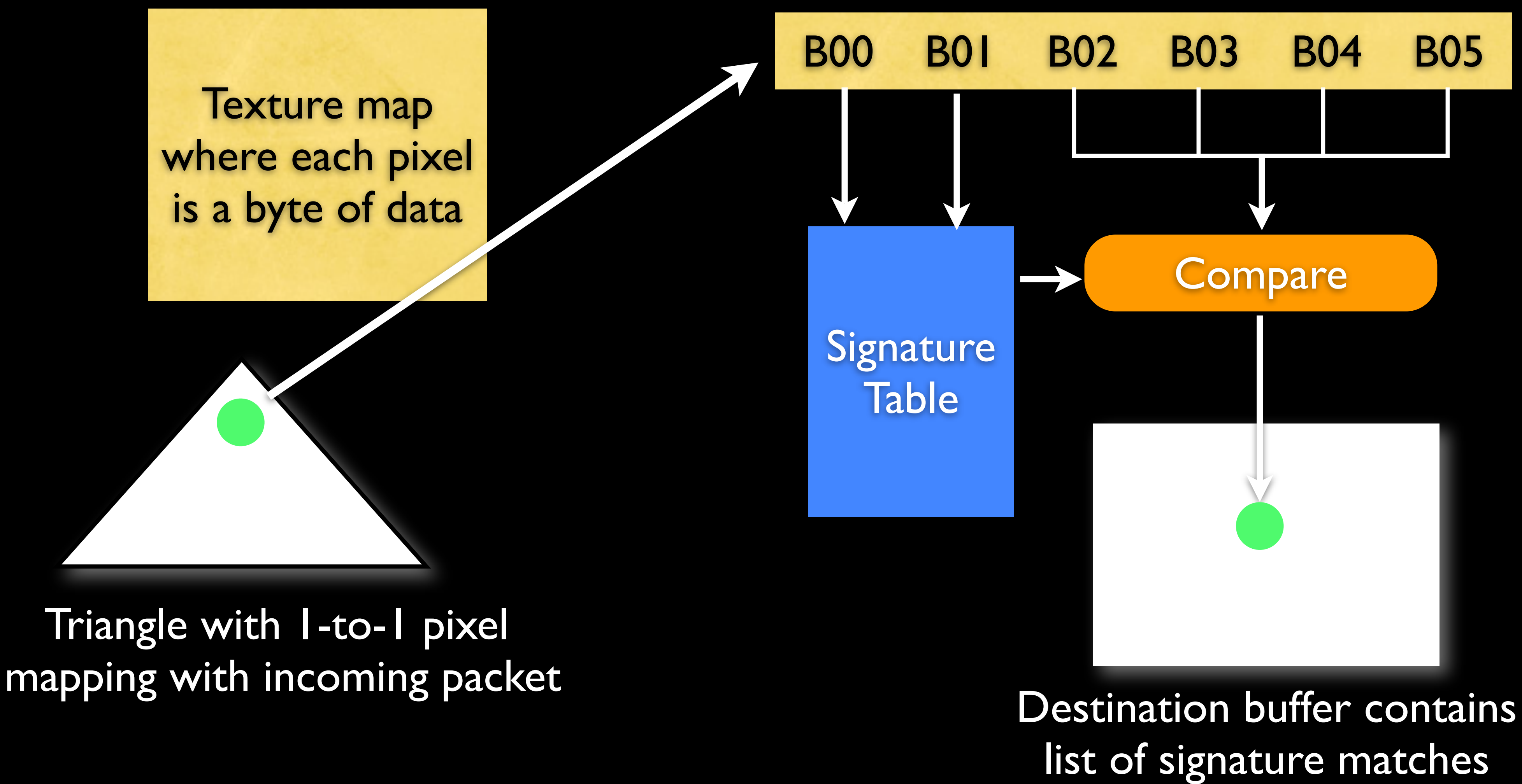
- Allows programmer to define how information is located on a grid
- i.e.
  - Precompute locations of elements to be processed
  - Draw 1 pixel points there and process in fragment shader



Example Application

- Virus Signature Matching
  - GPU architecture is very similar to specialist network processors
  - Pattern matching parallelizable over packets
  - GPU used for initial ‘potential’ mapping against large number of packets and CPU used for final verification

Virus Pattern Matching



Physically Based Modeling

Particle Systems

- Technique for modeling natural collection of objects using a collection of independent objects
- Each particle has its attributes as does the system which defines evolution over time
- Can be used for: fire, smoke, clouds, fog, explosions, grass

Particle Systems

- Particle Definition: A body whose spatial extent, internal motion and structure, if any, are irrelevant in a specific problem
- Usually either a 'super-atom' or a 'sample'

Particle System - Basics

- Typical particle attributes:
  - Position
  - Velocity
  - Life span
  - Size / Weight
  - Representation
  - Owning system

Particle System - Basics

- Particle system:
  - Associated particles
  - Emitters
  - Forces
  - State
  - Representation
  - Update function

Particle Systems - Calc

```
T = 0;
foreach step in time, Δt{
    T += Δt
    foreach particle {
        compute total force Fi on particle
        ai = Fi/mi
        vi += ai Δt
        xi += vi Δt
    }
    display()
}
```

Particle System - Display

- Have rendering rules which control how the system is displayed to look like the represented entity

Methods of representation

- Points - simplistic, limited
- Billboards
- Geometry - expensive
- Surface finding / aggregation

Billboards

- Single, textured quad that 'faces' user constantly
  - Either in spherical or cylindrical manner
- Texture can evolve over time
- Very cheap method of rendering particle systems

Modelview Matrix

$$\begin{matrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{matrix}$$

Sphere Billboard

$$\begin{matrix} 1 & 0 & 0 & a_{14} \\ 0 & 1 & 0 & a_{24} \\ 0 & 0 & 1 & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{matrix}$$

Cylinder Billboard

$$\begin{matrix} 1 & a_{12} & 0 & a_{14} \\ 0 & a_{22} & 0 & a_{24} \\ 0 & a_{32} & 1 & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{matrix}$$

Billboards cont.

- Matrices on previous slide produce Fig 1, more realistic is Fig 2
- Can get this with a rotate based on direction to camera:

1. normalize(objToCamProj)
2. angle = lookAt•objToCamProj
3. upVec = lookAt x objToCamProj
4. Rotate(angle,upVec)

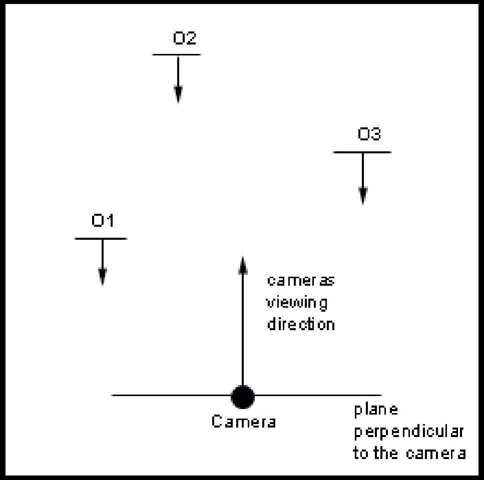


Fig 1

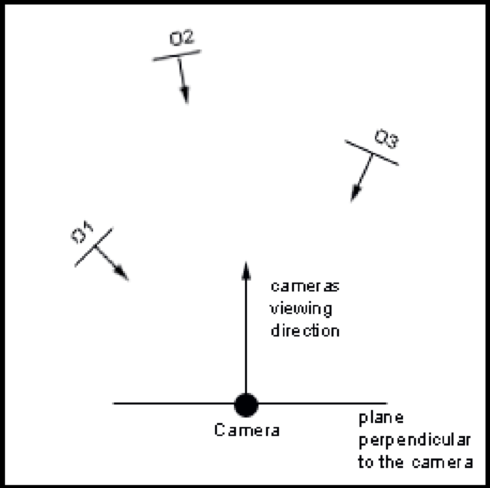


Fig 2

Differential Equation Solving

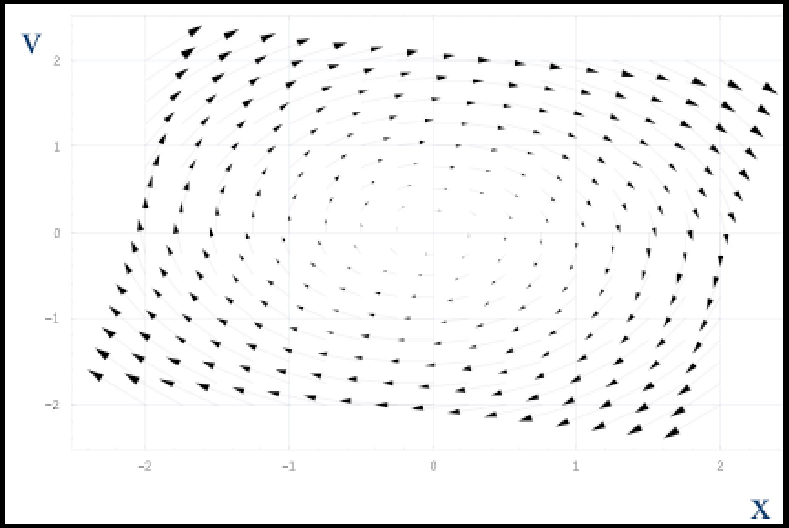
- Need to track the state of the system
- Is an example of an 'Initial Value Problem':

  $$\dot{x} = f(x, t)$$

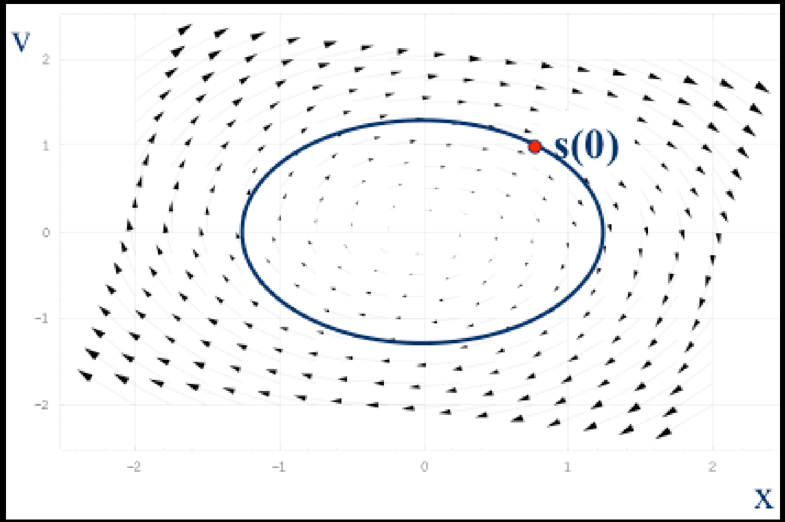
- Need to use an ODE to solve

Initial Value Problems

- Undamped motion as a velocity field:



- Tracking a particle in this field

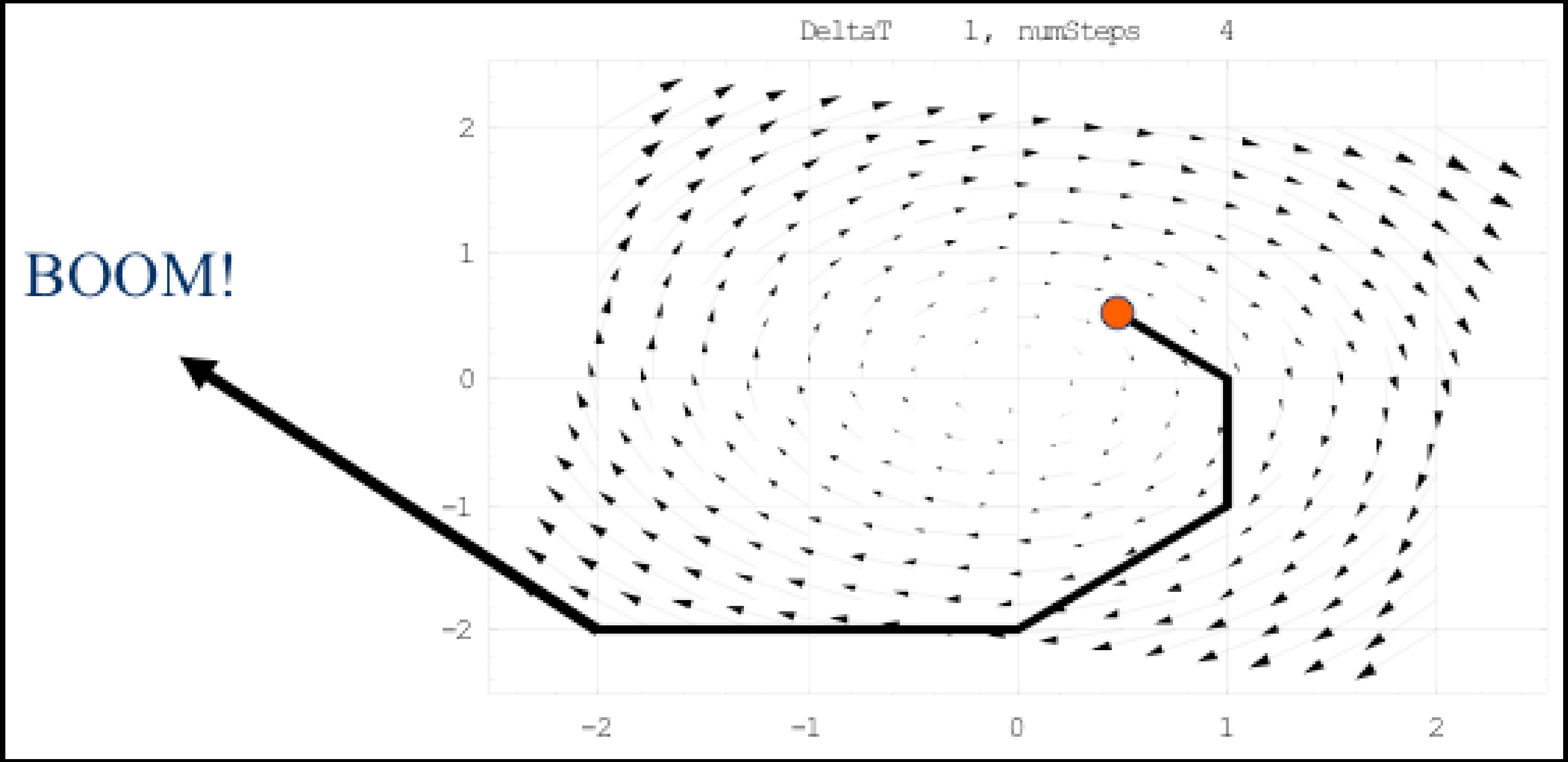


ODEs

- Euler:
  - Simplistic method of solution which takes 1 step between t and $t+\Delta t$
  - Need very small timesteps and/or large damping to stop ‘blow up’

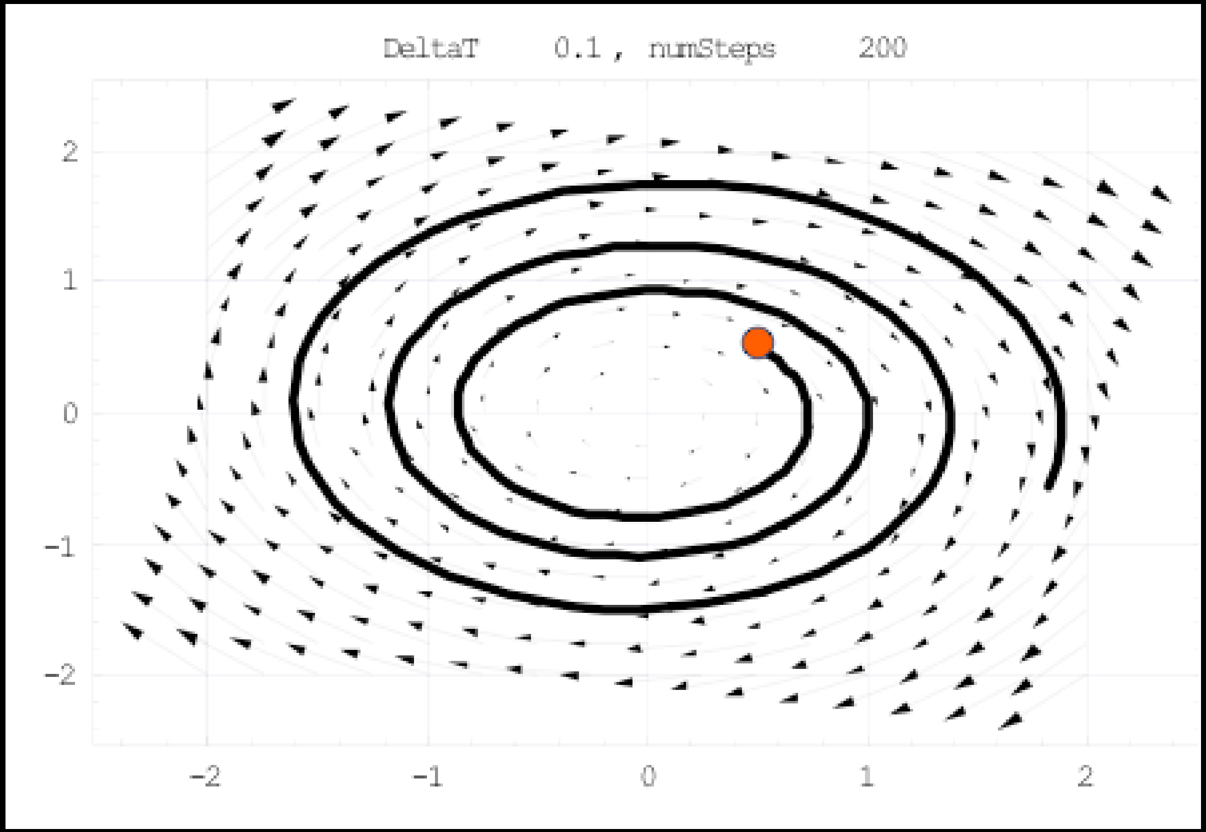
Problem with Euler

- Δt: 1 num steps: 4



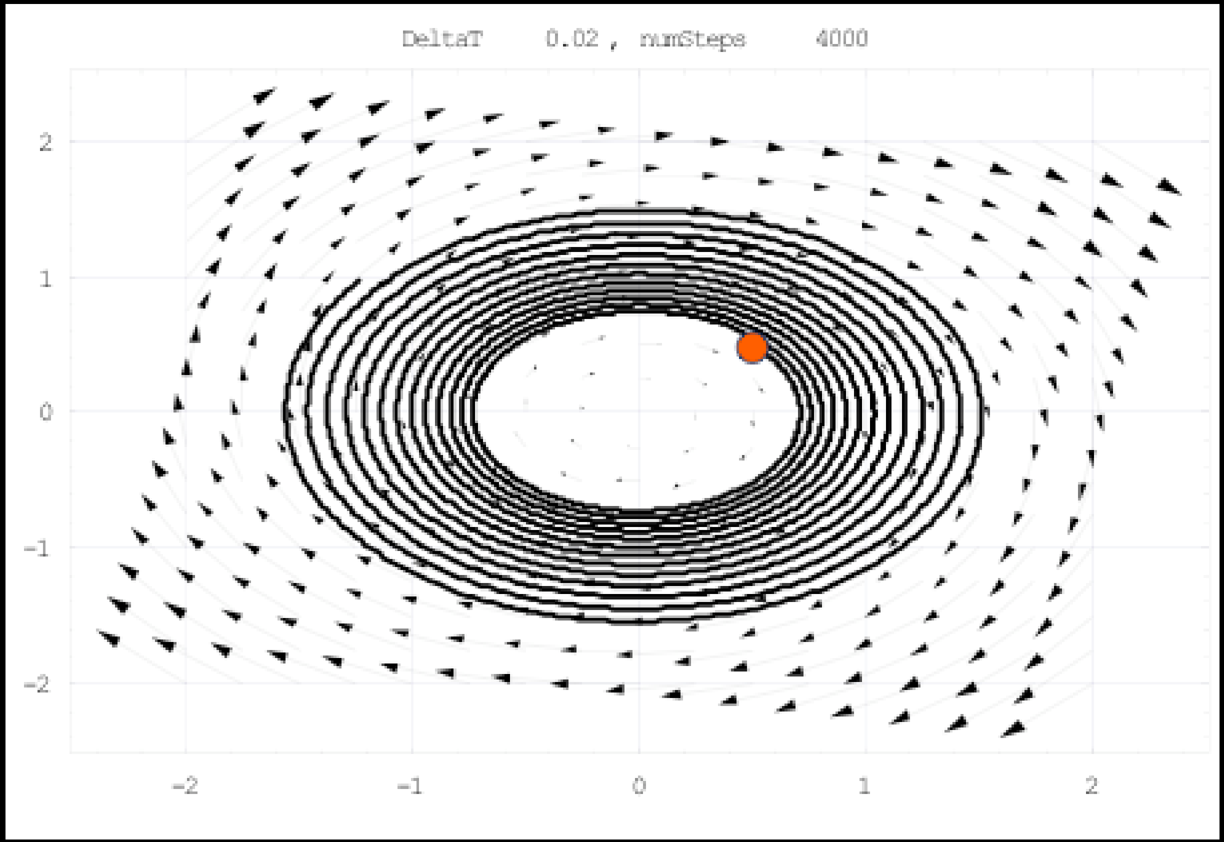
Problem with Euler

- Δt: 0.1 num steps: 200



Problem with Euler

- Δt: 0.02 num steps: 4000



Sources

- Harris, M (2006). GPGPU Lessons Learned. GDC 2006
- Lobb, R (2003). Physically Based Animation Lecture Notes. COMPSCI715 2003
- Nguyen, H (2007). GPU Gems 3. Addison-Wesley Professional